Guide to Text Processing on VAX/VMS

Order Number: AI-Y502B-TE

April 1986

This manual contains tutorial information about the EDT editor, the EDT Keypad Emulator and the EVE interface for VAXTPU, and DIGITAL Standard Runoff (DSR).

Revision/Update Information:	This revised document supersedes the *Guide to Text Processing on VAX/VMS* Version 4.0.
Software Version:	VAX/VMS Version 4.4

digital equipment corporation
maynard, massachusetts

April 1986

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The first example in Section 4.1 (What is DSR?) is reprinted from page 70 of *The Elements Of Style*, Third Edition, by William Strunk, Jr. and E.B. White with permission of the publisher. Copyright 1979 by Macmillan Publishing Company.

The last example in Section 4.10 (How to Create Running Heads) is reprinted from *Faberge Eggs* by permission of Harry N. Abrams, Inc., Publishers.

ZK-2831

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

Preface

Intended Audience

This manual is intended for the novice user of EDT, the EDT Keypad Emulator, EVE, and DSR.

Structure of This Document

This manual is divided into four chapters. Chapter 1 explains how to use the EDT editor. Chapters 2 and 3 explain how to use the EDT Keypad Emulator (Chapter 2) and the EVE interface (Chapter 3) for the VAX Text Processing Utility (VAXTPU). Chapter 4 explains how to use DIGITAL Standard Runoff (DSR).

Associated Documents

For detailed reference information about the EDT editor, see the *VAX EDT Reference Manual.* For detailed reference information about DSR, see the *VAX DIGITAL Standard Runoff Manual.* For detailed reference information about VAXTPU, the EVE interface, or the EDT Keypad Emulator, see the *VAX Text Processing Utility Reference Manual.*

Conventions Used in This Document

Convention	Meaning
RET	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, RET .
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O. In examples, this control key sequence is shown as ^x, for example, ^C, ^Y, ^O, because that is how the system echoes control key sequences.

Convention	Meaning
$ SHOW TIME 05-JUN-1985 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in the black letters. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, mean either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec, . . .	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

The *Guide to Text Processing on VAX/VMS* now includes chapters on the EDT Keypad Emulator and the EVE interface for VAXTPU.

1 Editing Files with EDT

1.1 Introduction to the EDT Editor

EDT is an interactive text editor. You can use EDT to edit many kinds of text files—letters, memos, or complex computer programs. With EDT you can create new files, insert text into them, and edit that text. You can also edit text in existing files.

EDT offers many features to make text editing easier and more efficient. These features include the following:

- Three types of editing: keypad mode, line mode, and nokeypad mode. You can move from one mode to another during the same editing session. Keypad mode is screen-oriented, which allows you to see several lines of text simultaneously and move the cursor throughout the text in any direction. Line mode enables you to edit text by using line numbers. You can use nokeypad mode commands to define keys.
- Online HELP. You can use HELP any time during your editing session without affecting your work.
- Journal facility. The journal file protects your editing work in case of a system interruption.
- Access to files and buffers. You can work with as many files and buffers as you need during your EDT session.
- Startup command files. These enable you to personalize the characteristics of your editing sessions.
- Key definition facility. You can define keys to automate your keypad editing work.
- EDT macros. You can create EDT macros to automate your editing work. Macros can be saved for use in later editing sessions.
- Tabbing facility. This feature enables you to create layered text formats.

The discussion of EDT in this handbook is designed for the novice as well as the more experienced EDT user. For more detailed information about the EDT editor, see the *VAX EDT Reference Manual* (included in your doc set) and the *EDT Editor Manual*, which you can order separately (order number AA-M476A-TK).

1.1.1 Invoking and Terminating EDT

An editing session begins when you invoke EDT with the DCL command EDIT and ends when you terminate EDT with the EXIT or QUIT command. You may start an editing session by creating a file and inserting the text for the file during the course of the session. Or, you may specify an existing file when you start the session. EDT does not destroy the contents of any existing file that you edit; it simply produces a new version, leaving the old version intact.

1.1.1.1 Invoking EDT

To invoke EDT, enter the DCL command EDIT. You will be prompted for the name of a file. The following example shows how you invoke EDT to edit a file named MEMO.TXT:

```
$ EDIT
_File: MEMO.TXT
```

If you are creating a new file, you will see the message "Input file does not exist" followed by the end of buffer sign [EOB] and the asterisk prompt (*). If you are editing an existing file, a copy of the first line of text appears on the screen followed by the asterisk prompt.

The following example demonstrates how to invoke EDT to create a new file:

```
$ EDIT NEWFILE.FUN
Input file does not exist
[EOB]
*
```

The following example demonstrates how to invoke EDT to edit a file that already exists.

```
$ EDIT OLDFILE.DAT
     1        This is the first line of the file.
*
```

Once you enter EDT, you can choose between three different modes:

1. Line mode—Line mode allows you to use line numbers during your editing session. Line mode commands are particularly useful for manipulating large blocks of text.
2. Keypad mode—Keypad mode allows you to move the cursor to the text you want to edit and perform most editing functions by pressing keypad keys. You insert text, in keypad mode, by typing from the main keyboard.

3 Nokeypad mode—Nokeypad mode allows you to move the cursor directly to the text you want to modify and enter nokeypad commands.

One difference between the three modes is that you press keys to edit text in keypad mode whereas you enter commands to edit text in line mode and nokeypad mode.

By default, EDT puts you into line mode. You will know that you are in line mode when you see the asterisk prompt (*). If you want to enter keypad mode, type the letter "C" (abbreviation for CHANGE command) when you see the asterisk prompt, and press RETURN. If you want to enter nokeypad mode, enter the SET NOKEYPAD command when you see the asterisk prompt, press RETURN, type the letter "C," and press RETURN again.

The following table shows the three modes of EDT and the commands you enter to invoke each mode on a video terminal.

Mode	Commands to Invoke Mode
Line	$ EDIT *
Keypad	$ EDIT * C
Nokeypad	$ EDIT * SET NOKEY * C

1.1.1.2 Terminating EDT

Both the EXIT and QUIT commands terminate an editing session; however, only EXIT saves your edits. When you enter the EXIT command, EDT creates an output file containing the edited version of the input file. By default, the output file will have the same name and type as the input file. (The version number is incremented by one.) The following example demonstrates how to invoke and terminate EDT to edit a file named FUN.DAT. The output file has the same name as the input file and the version number is increased by one.

```
$ EDIT FUN.DAT
   .
   .
   .
*EXIT
DBA2:[WHITEBRIDGE]FUN.DAT;5  2 lines
```

If you want to override the default, and specify a different output file name, enter the EXIT command and specify the new file name as a parameter. In the following example, you invoke EDT to edit a file named FUN.DAT and specify JOKE.DAT as the new file name when you terminate the session.

```
$ EDIT FUN.DAT
    .
    .
    .
*EXIT JOKE.DAT
DBA2:[WHITEBRIDGE]JOKE.DAT;1  2 lines
```

If you do not want to save your edits when you end an editing session, use the QUIT command. All the edits you made to the file will be lost and no output file will be created.

The following table shows the three modes of EDT and the commands you enter to terminate each mode.

Mode	Commands to Terminate EDT
Line	* EXIT or QUIT $
Keypad	CTRL/Z * EXIT or QUIT $ or * GOLD key, COMMAND key, EXIT or QUIT command, ENTER key $
Nokeypad	EX * EXIT or QUIT $ or QUIT $

1.1.2 Using the HELP Facility

EDT's online HELP facility allows you to get help during your editing session without interrupting your work.

In keypad mode you press the HELP key to get information about keypad functions. For information about line and nokeypad commands, you use the line mode command HELP. The following sections discuss the HELP facility in more detail.

1.1.2.1 Accessing HELP from Line Mode

You can type the HELP command after the asterisk prompt (*) for information about EDT commands. If you want information about specific commands, type the HELP command followed by the name of the command. For example, if you want to know more about the INSERT command, type the following:

```
*HELP INSERT
```

If you want information on a qualifier, for example /SAVE, type:

```
*HELP EXIT/SAVE
```

You must include the name of the command, in this case EXIT, and the slash (/) before the qualifier.

EDT can help you get information on commands even when you supply only the first letter or so. If you enter HELP D, EDT prints the information for both the DEFINE and DELETE commands. HELP CH gives you the CHANGE command, but HELP C supplies information on CHANGE, CLEAR, and COPY.

Use the wildcard character (*) with the HELP command to get information on all subtopics for a command without having to supply the subtopic names. For example, the following command prints information on TYPE /BRIEF and TYPE/STAY:

```
*HELP TYPE *
```

1.1.2.2 Accessing HELP from Keypad Mode

You can press the HELP key to get a diagram of the keypad functions. The screen goes blank for a few seconds before the diagram appears.

After EDT displays the diagram, you can press any keypad key to see a description of what the key does. If you press the keypad key numbered 5, for example, a description of the BACKUP and TOP keypad functions appears on the screen. At the bottom of each description are directions for seeing the whole keypad diagram again (by pressing RETURN), reading about other functions (by pressing another keypad key), or returning you to your editing session.

For help on a control key sequence such as CTRL/A, press both the control key and the keyboard key. For help on a GOLD key sequence, such as GOLD/A, press only the keyboard key, not the GOLD key.

Note

The HELP diagrams will not display any keyboard or keypad keys that you have redefined.

1.1.2.3 Accessing HELP from Nokeypad Mode

To get HELP information about nokeypad commands, type EXIT, press RETURN, and type HELP CHANGE. You will see five subtopics:

1 ENTITIES

2 HARDCOPY

3 KEYPAD

4 SCREEN

5 SUBCOMMANDS

The ENTITIES subtopic contains a list of the nokeypad entity subtopics: character, word, line, range, sentence, page, paragraph, select, vertical, and string. When you want information about one of these entities, for example sentence, include the entity name as the last HELP command subtopic:

```
*HELP CHANGE ENTITIES SENTENCE
```

The HARDCOPY, KEYPAD, and SCREEN subtopics provide brief descriptions of hardcopy, change mode, keypad mode, and screen editing. These do not have further subtopics.

Information on nokeypad commands is contained in the SUBCOMMANDS subtopic. For a complete list of the subtopics covered, type:

```
*HELP CHANGE SUBCOMMANDS
```

1.1.3 Recovering from Interruptions

If your EDT editing session is interrupted, EDT provides a way for you to recover your work by using a journal file. Journal files and the method for recovering from interruptions are discussed in Section 1.7.

1.1.4 Moving Between Modes

Once you become familiar with the three modes of EDT, you will want to travel between them. The following table lists the commands you need to know to move between the modes.

From	To	Command
Line Mode	Keypad Mode	*CHANGE
Line Mode	Nokeypad Mode	*SET NOKEYPAD
		*CHANGE
Keypad Mode	Line Mode	CTRL/Z
Keypad Mode	Nokeypad Mode	CTRL/Z
		*SET NOKEYPAD
		*CHANGE
Nokeypad Mode	Line Mode	EX
		*
Nokeypad Mode	Keypad Mode	EX
		*SET KEYPAD
		*CHANGE

The following example demonstrates how to invoke EDT to create a new file named FUN.FUN, enter line mode (by default), and enter keypad mode.

```
$ EDIT FUN.FUN ❶
Input file does not exist
[EOB]
*CHANGE ❷
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

The following example demonstrates how to invoke EDT to create a new file named WHY.NOT, enter line mode (by default), enter keypad mode, return to line mode, and finally enter nokeypad mode.

```
$ EDIT WHY.NOT ❶
Input file does not exist
[EOB]
*CHANGE ❷
[EOB]
CTRL/Z ❸
*SET NOKEYPAD ❹
*CHANGE ❺
```

❶ The EDIT command invokes EDT.

❷ The asterisk prompt (*) indicates that you are in line mode and the CHANGE command invokes keypad mode.

❸ CTRL/Z returns you to line mode.

❹ The asterisk prompt indicates line mode and the SET NOKEYPAD command invokes nokeypad mode.

❺ The CHANGE command invokes nokeypad mode (only after you enter the SET NOKEYPAD command).

1.2 How to Use Keypad Mode

Keypad editing is available on VT200 Series, VT100, and VT52 terminals. In keypad editing, the contents of a file are displayed on the screen as you edit. You can see the changes you make to a file as they take place.

In keypad editing, you press keys to perform editing functions rather than typing commands as is done in line and nokeypad editing.

1.2.1 VT200 Series, VT100, and VT52 Terminal Keypads

Figure 1–1 shows all the keypad functions available on VT100, VT52, and LK201 keyboards. (An LK201 keyboard is used with VT200 Series terminals.) Each key in the keypad performs at least one editing command; many of the keys perform two. You can use the standard function (on the upper half of the key) by pressing the key. You can use the alternate function (on the lower, shaded half of the key) by pressing the GOLD key before pressing the key.

Notice that the LK201 keyboard includes a numeric keypad (like the keypad on the VT100 terminal) and a six-key editing keypad with four arrow keys. (Two of the twenty available function keys, HELP and DO, are also displayed.) See the *VAX EDT Reference Manual* for more detailed information about the LK201 function keys.

1.2.2 Using the GOLD Key

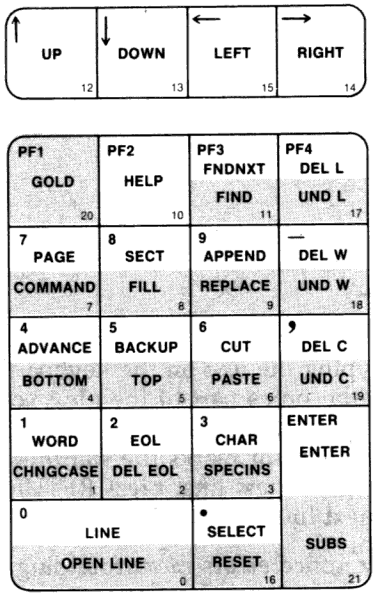
You can use the GOLD key for two purposes:

1 To use the alternate of two functions on a keypad key

2 To execute a function a specified number of times

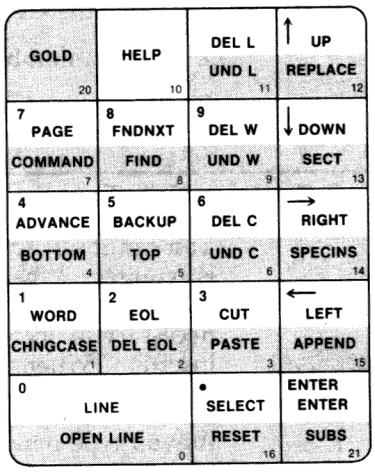
Figure 1–1 VT100, VT52, and LK201 Keypads

Keypad Editing Keys - VT100 Terminals



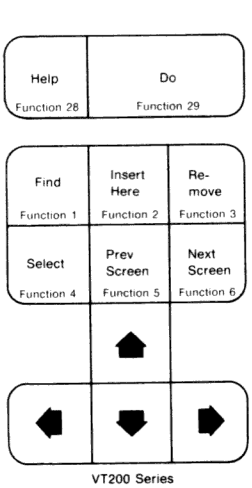
VT100

Keypad Editing Keys - VT52 Terminals

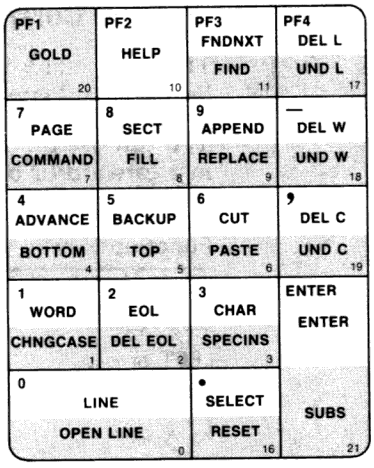


VT52

Keypad Editing Keys - LK201 Keyboard



VT200 Series



ZK-1605-84

To use an alternate function, press the GOLD key followed by the desired key. (You do not need to hold down the GOLD key.) For example, to use UND C, you press the GOLD key first, then the UND C key.

If you want to execute a function a specified number of times, for example, to use DEL C twelve times, you press the GOLD key, type the number 12 (on the main keyboard), and press DEL C. Try it and see how much time you save. The following steps show how to repeat any keypad function a specified number of times:

1. Press the GOLD key.
2. Type a number.
3. Press the desired function.

1.2.3 Inserting Text

You can insert text simply by typing the text on the keyboard. No command is required. Whatever you type becomes part of the file. Your insertion appears on the screen as you type it and the surrounding text moves as necessary to accommodate it. The cursor marks the starting location for the insertion. When you want to begin a new line, press RETURN to move the cursor to the beginning of the next line and continue typing your text.

As you type new text, you may notice errors in surrounding text. You can move the cursor to these errors and correct them at any time, and then move the cursor back and continue to insert text.

1.2.4 Moving the Cursor

There are many ways to move the cursor in keypad mode. The following lists several keypad functions that move the cursor.

You can use the ADVANCE and BACKUP keypad functions to set the cursor in a forward or backward direction, respectively.

Function Key	Destination of Cursor
TOP	Beginning-of-Buffer
BOTTOM	End-of-Buffer
LEFT arrow	One character to the left
UP arrow	Up one character

Function Key	Destination of Cursor
RIGHT arrow	One character to the right
DOWN arrow	Down one character

The following example demonstrates how to move the cursor to the beginning (or top) and to the end (or bottom) of the buffer using the TOP and BOTTOM keypad functions.

Insert the following three lines of text:

```
Five golden rings,
four calling birds,
three french hens,
```

1. Press the GOLD key followed by TOP to move the cursor to the "F" in the word "Five."
2. Press the GOLD key followed by BOTTOM. Notice the cursor moving to the left bracket of the end-of-buffer sign [EOB].
3. Press the GOLD key followed by TOP again, moving the cursor back to the "F."

Repeat these steps to become familiar with the TOP and BOTTOM keypad functions.

The following example demonstrates how to move the cursor using the four arrow keys: UP, DOWN, LEFT, RIGHT.

```
Under a toadstool crept a wee elf,
Out of the rain to shelter himself.
Under a toadstool all in a heap,
Sat a big doormouse, fast asleep.
```

1. Press the TOP keypad function (the GOLD key followed by TOP) to move the cursor to the letter "U" in the word "Under."
2. Press the RIGHT arrow 9 times, then press the DOWN arrow once, moving the cursor to the "e" in the word "the."
3. Press the LEFT arrow 5 times, then press the DOWN arrow once, moving the cursor to the "r" in the word "under."
4. Press the RIGHT arrow 8 times, then press the DOWN arrow once, moving the cursor to the second "o" in the word "doormouse."
5. Press the LEFT arrow 6 times, then press the UP arrow 3 times, moving the cursor to the word "a."

The following example demonstrates the keypad functions for:

TOP
LINE
EOL
WORD
CHAR

Enter the following four lines of text:

```
ONE TWO THREE FOUR FIVE
SIX SEVEN EIGHT NINE
TEN ELEVEN TWELVE THIRTEEN
FOURTEEN
```

1. Press TOP to move the cursor to the first letter of the word "ONE."
2. Press LINE once, moving the cursor to the "S" in the word "SIX."
3. Press WORD twice, moving the cursor to the "E" in the word "EIGHT." Then press EOL. The cursor will move to the end of the line.
4. Press WORD three times, moving the cursor to the "T" in the word "TWELVE." Then press EOL. The cursor will move to the end of the line.
5. If you are using a VT200 Series or VT100 terminal, press CHAR four times. The cursor will move to the "R" in the word "FOURTEEN."

The following figure shows the approximate distances that some keypad commands move the cursor.

Keypad Function	Distance Keypad Function Moves Cursor
CHAR	#
WORD	######
EOL	###########
LINE	##
SECT	## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##

Keypad Function	Distance Keypad Function Moves Cursor
PAGE	## ########################################

Note that the PREV SCREEN and NEXT SCREEN keys (on an LK201 keyboard) work like the SECT keypad function, moving the cursor back or ahead by 16 lines.

1.2.4.1 How EDT Views Words

You can determine the way in which EDT recognizes a word, sentence, paragraph, or page. By default, EDT identifies a word as any group of contiguous characters starting at the current cursor position and continuing in the current direction until one of the following characters is encountered:

- Space ( )
- Horizontal tab (TAB)
- Linefeed (<LF>)
- Vertical tab (<VT>)
- Form feed (<FF>)
- Carriage return (<CR>)
- Line terminator

For example, enter the following text and move the cursor throughout the paragraph by pressing the WORD key. Notice that the cursor stops on the first letter of each word.

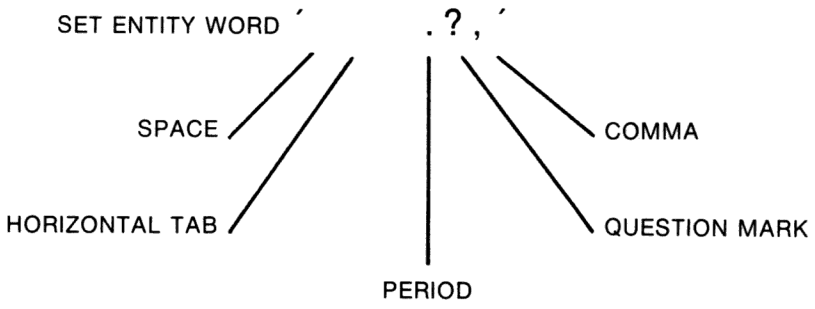
```
This is the first line of text.
Is this the second?
This must be the third.
How about a fourth line?
And, a fifth, perhaps?
Will anyone take a sixth?
Number seven, please.
```

You can use the SET ENTITY WORD command to change the way EDT views words. If you want EDT to recognize periods (.), question marks (?), and commas (,), enter the following command line in your startup command file.

```
SET ENTITY WORD ".?,"
```

See Figure 1.2. For information about startup command files, see Section 1.11.

Figure 1–2 Using the SET ENTITY WORD Command



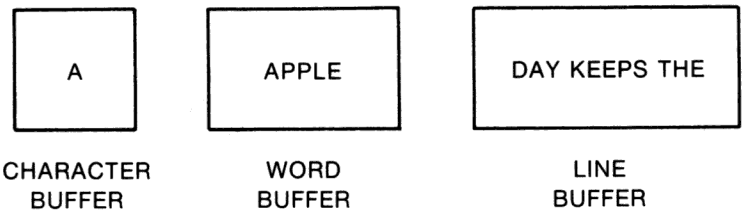
When you use the WORD key to move throughout the text again, notice that the cursor stops on all periods, question marks, and commas. For more information about the SET ENTITY command, see the *VAX EDT Reference Manual*.

1.2.5 Deleting and Undeleting Text

You can delete text by character, word, and line. DELETE and DEL C delete characters. DEL W and LINEFEED delete words or parts of words. DEL L, DEL EOL, and CTRL/U delete lines or parts of lines. The deleted text is stored in one of three buffers so that you can restore it using the UND commands (UND C, UND W, UND L). (A buffer is a temporary holding area for text. See Section 1.6 for more information on buffers.) The character buffer contains the last character deleted; the word buffer contains the last word deleted; and the line buffer contains the last line deleted. Only the most recent deletion can be restored; you can restore this unit of text numerous times to any location.

Figure 1–3 shows the three buffers that EDT fills when you use the delete and undelete keys.

Figure 1–3 Three EDT Buffers Used for Deleting and Undeleting Text



You can use the following keypad functions to delete text:

Keypad Function	What It Does
DELETE	Deletes the preceding character
DEL C	Deletes the current character
DEL W	Deletes to the end of the word
LINE FEED	Deletes to the beginning of the previous word
DEL L	Deletes to the next line
CTRL/U	Deletes from the beginning of the line to the cursor
DEL EOL	Deletes from the cursor to the end or beginning of the line, depending on the current cursor direction
CUT	Deletes the selected string from the current file and stores it in the PASTE buffer. The selected string is all the text between the cursor location when you pressed the SELECT keypad function and the position to which you moved the cursor. (Note that the REMOVE key on the LK201 keypad works like the CUT keypad function.)

Enter the following three lines of text. Then follow the steps to learn how to use the delete keypad functions.

```
CHARACTER
WORD WORD WORD WORD WORD
LINE LINE LINE LINE LINE LINE LINE LINE
```

1 Move the cursor to the first C in the word "character." Press DEL C. The C will disappear.

2 Press the GOLD key followed by UND C. The C will reappear.

3 Move the cursor to the H. Press DEL C to make it disappear.

4 Then press the GOLD key followed by UND C. The H is restored.

5 Press the GOLD key followed by UND C again, another H will appear.

Notice that the DEL C keypad function deletes the character directly at the cursor and the DELETE key (on the main keyboard) deletes the character immediately to the left of the cursor. You can use the UND C keypad function to restore the most recently deleted character in either case. Press the DELETE key and the DEL C keypad function to see the difference between the two.

1 Move the cursor to the W in "WORD." Press DEL W. The word "WORD" will disappear.

2 Press the GOLD key followed by UND W to restore the word.

3 Press the GOLD key followed by UND W again, another "WORD" will appear.

Notice that the DEL W keypad function deletes to the end of the current word, and the LINE FEED key (on the main keyboard) deletes to the beginning of the preceding word. Press the LINE FEED key to see the difference between the DEL W keypad function and the LINE FEED key. Again, you can use the UND W keypad function to restore the deleted word in either case.

1 Move the cursor to the beginning of the line of LINES. Press the DEL L keypad function. The entire line will disappear.

2 Press the GOLD key followed by UND L to restore the line.

3 Press the GOLD key and UND L several times to create multiple lines.

Use the CTRL/U command to delete text from the cursor to the beginning of the line. Notice that the cursor moves to the beginning of the line. Press UND L to restore the line.

1.2.6 Locating Text

Use the FIND and FNDNXT keypad functions to locate strings of text. When you press the GOLD key followed by FIND you will see the following prompt on the screen:

```
Search for:
```

Type the string of text you are looking for and press one of the following keys:

- ADVANCE
- BACKUP
- ENTER
- DO

If you press ADVANCE, EDT will search in a forward direction for the specified string. If you press BACKUP, EDT will search in a backward direction. If you press ENTER or DO, EDT will search in the direction already set.

Note

The ADVANCE and BACKUP keys set the direction for subsequent EDT commands.

Enter the text below. Press the GOLD key followed by the FIND keypad function. When you are prompted for a search string, enter the word "ticket." Then press BACKUP. The search will start at the end of the text (where the cursor is located) and continue towards the beginning of the text until it finds the word "ticket."

```
Ella will not be able to attend the concert tonight.
She has a sore throat.  Perhaps you could give the ticket
to somebody in your music class.  Ella wants to see the
program when she is feeling better.
Search for: ticket
```

Now search for the word "program." Press the GOLD key followed by FIND. When you are prompted for the search string, enter the word "program." If you press BACKUP again, you will see the response "String was not found" because the word "program" is located after the word "ticket." Try it. Now press ADVANCE followed by FNDNXT. EDT will find the string.

```
Ella will not be able to attend the concert tonight.
She has a sore throat.  Perhaps you could give the ticket
to somebody in your music class.  Ella wants to see the
program when she is feeling better.
Search for: program
```

Remember the following three items when you are entering a search string:

1 EDT ignores diacritical marks and the case of letters while making searches (unless you enter the SET SEARCH EXACT command).

2 DELETE is the only key you can use to edit an incorrectly typed search string.

3 To cancel a search string, press CTRL/U.

If you want to find the next occurrence of the string you are searching for, use the FNDNXT keypad function. EDT will search in the direction already set. If EDT cannot find the string, it will give you the message "String was not found." You can reverse the direction of the search by pressing either ADVANCE or BACKUP before pressing FNDNXT.

Enter the text below. Press GOLD and FIND. Enter the search string "little" when prompted. Because the cursor is at the end of the text, you must press the BACKUP keypad function to set the search in a backward direction. Now press FNDNXT nine times. EDT will find each occurrence of the search string. When the cursor arrives at the first "little," press the ADVANCE keypad function to reverse the direction of the search. Every time you press FNDNXT, the cursor will move forward to the next occurrence of the word "little."

```
One little, two little, three little chickadees,
four little, five little, six little chickadees,
seven little, eight little, nine little chickadees,
ten little chickadees feeding.
Search for: little
```

1.2.7 Moving Text

You can move text using three different groups of keypad functions:

1 RETURN and OPENLINE

2 DEL L, UND L, DEL W, UND W, DEL C, and UND C

3 CUT and PASTE (or REMOVE and INSERT HERE)

Using the first method, you press RETURN to move text (with the cursor) and OPENLINE to move text (without the cursor) down the screen line by line. Using the second method, you delete a unit of text (character, word, or line) from one location, move the cursor to another location, and undelete the text there. (See Section 1.2 for more information about deleting and undeleting text.) You can use the third method, which is described in this section, to move larger units of text.

Use the following three keypad functions to move text:

1 SELECT

2 CUT (or REMOVE)

3 PASTE (or INSERT HERE)

Press SELECT to mark one end of a string of text that you want to delete or move. Then move the cursor either backwards or forwards, to the other end of the string and press CUT. Before you press CUT, you can correct a mistake by pressing the GOLD key followed by RESET to cancel the select range and start over.

You press CUT to delete the selected string of text from the current file and store it in the PASTE buffer. (See Section 1.6 for more information about the PASTE buffer.) The selected text is all the text between the cursor location when you pressed SELECT and the position to which you moved the cursor.

If you press CUT before you press SELECT you will see the message "No select range active."

You press GOLD followed by PASTE to insert the contents of the PASTE buffer to the left of the cursor position.

To summarize, use the following steps to move text:

1 Place the cursor at the beginning of the text you want to move.

2 Press SELECT to mark the location.

3 Move the cursor to the end of the select range.

4 Press CUT to delete the text from its current position.

5 Move the cursor to the character just beyond where you want the text inserted.

6 Press the GOLD key followed by PASTE.

Note

If you want to restore the select string to its original location after you perform Step 4 above, press the GOLD key followed by PASTE.

Enter the following text:

```
january february march april
may june july november
december august september october
```

In order to move the string "august september october" after "july," move the cursor to the "a" in "august." Press SELECT. Then move the cursor to the "r" in "october." Press CUT. The selected string will disappear into the PASTE buffer. Now move the cursor after the word "july" and press PASTE. A copy of the selected string in the PASTE buffer will appear on the screen between the words "july" and "november."

If you press PASTE again, you will get another copy of the contents of the PASTE buffer. Try it. Every time you press PASTE, you will get another copy of the string "august september october." Once you perform another SELECT and CUT operation, specifying a different string, the contents of the PASTE buffer will change. You can always get as many copies of the PASTE buffer as you want by pressing PASTE.

If you want to add to the text in the PASTE buffer, retaining the text already there, use APPEND. APPEND deletes the select range from the current buffer and adds it to the end of the PASTE buffer.

For example, if your PASTE buffer contains the text "Wolfgang Amadeus" and you want to add the text "Mozart," follow the steps below:

1 Press SELECT.

2 Type the word " Mozart." (Precede the word with a space.)

3 Press APPEND.

Now the PASTE buffer contains the text:

```
Wolfgang Amadeus Mozart
```

1.2.8 Substituting Text

You can use the SUBS keypad function or the REPLACE keypad function to replace one string of text with another. The following four steps demonstrate how to use SUBS to make substitutions:

1 Press SELECT and enter the replacement text.

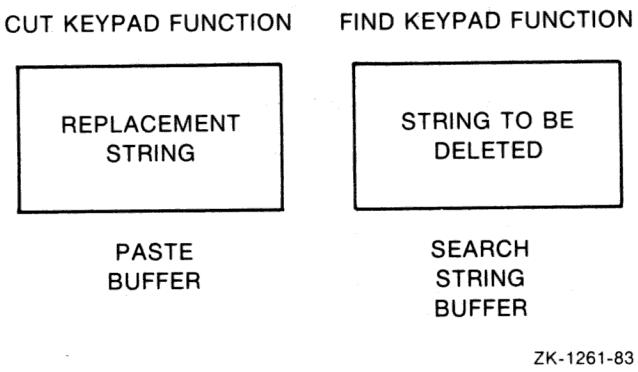
2 Press CUT. (The select range will disappear into the PASTE buffer.)

3 Press the GOLD key followed by FIND and enter the search string.

4 Press SUBS to exchange the existing text for the replacement text.

When you make substitutions using SUBS, you are using both the PASTE buffer and the search string buffer. You use CUT to put the replacement string in the PASTE buffer and FIND to put the string you want to find and delete in the search string buffer. Figure 1–4 shows both buffers.

Figure 1–4 Two EDT Buffers Used for Substituting Text



When you use SUBS, EDT performs the substitution first and then moves to the next occurrence of the search string. If EDT cannot find another occurrence of the search string, it prints the message "String was not found." If you do not want to make a particular substitution, press the FNDNXT keypad function. The cursor will move to the next occurrence of the search string. If you want to change that one, press SUBS again. Thus, you can move through the buffer pressing SUBS each time you want to make a replacement and pressing FNDNXT when you want EDT to leave the search string alone.

Note

You must use the FIND keypad function when you are making substitutions with SUBS because you are replacing text that matches the search string with the contents of the PASTE buffer. SUBS will not work correctly if you do not use FIND.

Enter the following text:

```
Susanne grabbed the red apple and gobbled it down.  Suddenly
her face turned quite red.  Glancing towards the red house
she saw the huge tree bathed in red leaves.
```

Now perform the following steps to substitute the word "green" for the word "red:"

1. Press SELECT and enter the word "green."
2. Press CUT. (The word "green" will disappear into the PASTE buffer.)
3. Press GOLD followed by FIND and enter the word "red."

4 Press SUBS to exchange the word "red" for the word "green."

5 Press SUBS three more times.

Your resulting text should look like this:

```
Susanne grabbed the green apple and gobbled it down.  Suddenly
her face turned quite green.  Glancing towards the green house
she saw the huge tree bathed in green leaves.
```

REPLACE deletes the text in the select range and replaces it with the contents of the PASTE buffer.

For example, if your PASTE buffer contains the words "paste paste paste," and your select range contains the words "select select select," press REPLACE to make your select range contain the words "paste paste paste." The following steps demonstrate this example:

1 Press SELECT.

2 Type the words "paste paste paste."

3 Press CUT. Now your PASTE buffer contains the words "paste paste paste."

4 Press SELECT again.

5 Type the words "select select select."

6 Press REPLACE. Now your select range contains the words "paste paste paste."

1.2.9 Five More Keys to Use with the GOLD Key

Five keypad functions associated with the GOLD key are summarized in the following table:

Keypad Function	What It Does
COMMAND	Enables you to enter a line mode command from keypad mode.
CHNGCASE	Reverses the case of letters in your text. Uppercase letters become lowercase; lowercase letters become uppercase.
FILL	Reorganizes the select range so that the maximum number of whole words can fit within the current line width.

Keypad Function	What It Does
RESET	Changes the following conditions of your editing session: • Cancels an active select range. • Empties the search buffer so that there is no current search string. • Sets EDT's current direction to ADVANCE. • Sets EDT to the default DMOV state. (DMOV is a Nokeypad command that returns your editing session to EDT's default state, where EDT does not alter the case of letters during move operations.)
SPECINS—special insert	Enables you to insert any character from the DEC Multinational Character Set into your text using the character's decimal equivalent value. For information about the DEC Multinational Character Set, see the *VAX EDT Reference Manual.*

For more information about all the available keypad keys, see the *VAX EDT Reference Manual.*

1.3 How to Use Line Mode

You can use EDT's line editing facility with any interactive terminal—hardcopy or screen. Line editing uses the line as its point of reference. EDT moves through the text line by line, not character by character as in the two other editing modes. Line editing commands are particularly useful for manipulating large blocks of text.

1.3.1 Line Numbers

To help you locate and edit text, EDT assigns line numbers. These line numbers are not part of the text and are not kept when you end an editing session. To see the line numbers of an already existing file, enter the TYPE WHOLE command after the asterisk prompt (*). You will notice that the text you enter is indented 12 spaces.

The following example demonstrates how to display your line numbers:

```
*TYPE WHOLE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    5           fivefivefivefive
[EOB]
*
```

Line numbers have the following characteristics:

- They are assigned to every line in every buffer in every editing session, including newly inserted lines and text added with the INCLUDE command.
- They start with 1 and increment by 1, unless otherwise specified with the RESEQUENCE/SEQUENCE command.
- They are decimal numbers if newly inserted.
- They are removed by the EXIT command, unless otherwise specified.
- They can be renumbered in increments of 1 or more with the RESEQUENCE command.

When you insert new text, EDT numbers the lines using decimal numbers. For example, if you add a line of text between lines 13 and 14, it is numbered 13.1. To avoid any visual confusion when working with decimal numbers, use the RESEQUENCE command to change the sequence of line numbers. When you type RESEQUENCE, EDT renumbers all the lines from the cursor to the end of the buffer in increments of 1. You can specify a range of lines within the file by entering a range after typing RESEQUENCE.

If you insert text after the fourth line in the following example and enter the TYPE WHOLE command again, you will see the newly inserted text preceded by decimal numbers. To change the sequence of line numbers, enter the RESEQUENCE command.

```
*TYPE WHOLE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    4.1         four_point_one
    4.2         four_point_two
    4.3         four_point_three
    5           fivefivefivefive
[EOB]
*RESEQUENCE
    1           oneoneoneoneoneoneone
    2           twotwotwotwotwo
    3           threethreethree
    4           fourfourfourfour
    5           four_point_one
    6           four_point_two
    7           four_point_three
    8           fivefivefivefive
[EOB]
*
```

The /SEQUENCE qualifier determines the increments EDT uses to number lines. When you type the following command and enter two numbers separated by a colon (n:m), EDT numbers the lines in the buffer, assigning number n to the first line and incrementing by number m.

```
*RESEQUENCE/SEQUENCE:
```

The following example resequences the contents of the buffer, assigning number 6 to the first line and incrementing by 4:

```
*RESEQUENCE/SEQUENCE:6:4 RET
3 lines resequenced
*TYPE WHOLE RET
    6           Washington, Maine
    10          Vermont, New York
    14          Ohio, New Mexico
    18          Colorado, Virginia
    22          Florida, Arizona
    26          Alabama, Oregon
    30          Minnesota, New Hampshire
    34          Texas, California
    38          Delaware, Michigan
[EOB]
*
```

1.3.2 Inserting Text

When you want to insert text, enter the INSERT command after the asterisk prompt (*), and press RETURN. The cursor will indent 16 spaces and wait for you to start typing. (Indenting is on the screen or paper only and does not become part of the edited text.) You can enter as many lines as you want. Each time you come to the end of a line, press RETURN to move to the beginning of the next line.

As you enter a line, you can delete characters by pressing the DELETE key. You can delete a portion of a line, from the cursor to the left margin, by entering CTRL/U. Press CTRL/Z to return to the line editing prompt. However, when you end a line with RETURN, you can no longer delete characters from that line in insert mode. (See Section 1.3 for more information about deleting text.) When you finish entering text, press CTRL/Z. EDT will echo ^Z on the screen. To display the text you just inserted, enter the TYPE WHOLE command after the asterisk prompt.

Note

EDT, which inserts text in front of the current line, is different from many other text editors that insert text following the current line.

The following example demonstrates how to insert text.

```
$ EDIT NEWFILE.DAT
Input file does not exist
[EOB]
*INSERT
                This is the first line that I would like
                to type.  At the moment there is no file
                named newfile.dat in my directory.  When
                I finish my editing session and type the
                EXIT command, EDT will save a copy of
                this text in my directory under the name
                newfile.dat.
[EOB]
*EXIT
DBAO:[GLOVER]NEWFILE.DAT;1 5 lines
```

1.3.3 Ranges

When you want a line mode command to affect a specific part of the buffer, you must enter a range. Ranges can specify one or more lines. In addition, the multiple-line ranges can be contiguous or noncontiguous.

The following table lists and describes the different ranges you can specify when editing in line mode.

Range Type	Description
period(.)	current line
number	EDT line number
'string'	next line containing the quoted string
BEGIN	first line of the buffer
END	after the last line in the buffer ([EOB])
LAST	last line EDT was at in the previous buffer
WHOLE	entire buffer
BEFORE	all lines in the buffer before the current line
REST	all lines in the buffer starting with the current line and ending with the last line

The following examples demonstrate each range type listed above.

EXAMPLES

1
```
TYPE
```

This command displays the current line.

2
```
TYPE 35
```

This command displays line 35.

3
```
TYPE "December"
```

This command displays the first line it encounters containing "December."

4
```
TYPE BEGIN
```

This command displays the first line of the current buffer.

5
```
TYPE END
```

This command displays the [EOB] mark.

6 TYPE LAST

This command displays the most recent line of text you displayed. The LAST specifier can only be used by itself; EDT does not accept LAST with other range specifiers or symbols or with buffer names.

7 TYPE WHOLE

This command displays every line in the current buffer.

8 TYPE BEFORE

This command displays the group of lines in the current buffer starting with the first line and ending with the line just before the current line.

9 TYPE REST

This command displays the group of lines in the current buffer starting with the current line and ending with the last line in the buffer.

You can use the following symbols and words with the range types listed above while editing in line mode.

Symbol/Word	Description
, or AND	Used to join noncontiguous ranges in a list; only single lines can be joined in this way
: or THRU	Indicates a group of lines starting with the first range specifier and ending with the second
n	Indicates the number of lines from the current line
# n or FOR n	Indicates the next "n" number of lines
+ "string" or "n"	Indicates that "string" or "n" refers to a line or lines after the current line
– "string" or "n"	Indicates that "string" or "n" refers to a line or lines before the current line
ALL "string" or "n"	Indicates that the command applies to all lines containing "string"

The following examples demonstrate the symbols and words listed above.

EXAMPLES

1 `TYPE 3,6,8`

This command displays lines 3, 6, and 8.

2 `DELETE 4 AND 13`

This command deletes lines 4 and 13.

3 `TYPE 25:"Hartford"`

This command displays the group of lines starting with line number 25 and ending with the line containing "Hartford."

4 `TYPE . THRU 150`

This command displays all the lines in the group starting with the current line and stopping with line number 150.

5 `DELETE 4#3`

This command deletes line 4 and the three lines following line 4.

6 `TYPE . FOR 10`

This command displays the current line and the next ten lines; the number ten refers to the tenth line after the current line.

7 `TYPE BEGIN +6`

This command displays the seventh line of the current buffer.

8 `DELETE -3 THRU`

This command deletes the current line and the three lines preceding it; the number –3 refers to the third line before the current line.

9 `TYPE "Dear"+2 THRU "Sincerely"-2`

This command displays the body of the letter, starting with the first paragraph and ending with the last.

10 `TYPE . THRU 1000 ALL "Date:"`

This command displays every line containing the string "Date:" that appears in the group of lines starting with the current line and ending with line number 1000.

1.3.4 Deleting Text

You can delete individual lines or groups of lines by using the DELETE command. After a delete operation, EDT displays the line following the last line deleted; this line is the new current line.

Use the following syntax:

```
*DELETE range
```

The following example demonstrates how to delete line 2 in a file named FUN.DAT.

```
$ EDIT FUN.DAT
    1           This is the first line of a file named FUN.DAT.
* TYPE WHOLE
    1           This is the first line of a file named FUN.DAT.
    2           This is the second.
    3           This is the third,
    4           and the fourth.
* DELETE 2
1 line deleted
    3           This is the third,
*TYPE WHOLE
    1           This is the first line of a file named FUN.DAT.
    3           This is the third,
    4           and the fourth.
*
```

If you enter the DELETE command and do not specify a range, EDT deletes the current line.

When you delete a range of lines, EDT deletes the lines and displays a message stating the number of lines deleted. The message is followed by a display of the next line in the text buffer.

You can use the /QUERY qualifier if you want EDT to prompt you before deleting each line of a specified range. The prompt is a question mark (?). Use one of the following four responses:

- Y (Yes) Delete this line.
- N (No) Do not delete this line.
- A (All) Delete all remaining lines in the specified range.
- Q (Quit) Quit the delete operation.

You must enter one of these responses before EDT continues to the next line or exits from the delete operation. The following example demonstrates the /QUERY qualifier with the DELETE command:

```
*DELETE 7 THRU 13/QUERY
    7           This is the first line of the range.
?N RET
    8           This is the second line of the range.
?Y RET
1 line deleted
    9           This is the third line.
?A RET
5 lines deleted
    14          This line is not in the range.
*
```

1.3.5 Substituting Text

When you want to substitute one string for another, use either the SUBSTITUTE or SUBSTITUTE NEXT command. These are the only line editing commands that can alter text within a line, as opposed to changing the entire line. The SUBSTITUTE command operates on the current line or on a specified range. It has the capacity to make a global substitution; that is, you can replace every occurrence of one string with another by using only one line command. The SUBSTITUTE NEXT command makes a substitution at the next occurrence of the string within the buffer.

The syntax for the SUBSTITUTE command is:

```
*SUBSTITUTE /old-string/new-string/[range] [/QUERY]
```

To substitute the string GOOD for BAD throughout a buffer, type the line command SUBSTITUTE, the old-string, and the new-string, separating all three with the same delimiter. (A delimiter is a character that marks the beginning or end of a string.) To apply the command to the entire buffer, specify WHOLE as the parameter. When the operation completes, EDT supplies a message indicating how many substitutions were made.

Enter the following text:

```
He felt bad.  He had had a bad time in a bad part of the city.
How did he know that all the stores would have bad merchandise.
So, he had bad luck trying to find the perfect gift.  His
stomach felt bad after a bad lunch.
```

Enter the following command:

```
*SUBSTITUTE/BAD/GOOD/WHOLE
```

The resulting text will look like this:

```
He felt GOOD.  He had had a GOOD time in a GOOD part of the city.
How did he know that all the stores would have GOOD merchandise.
So, he had GOOD luck trying to find the perfect gift.  His
stomach felt GOOD after a GOOD lunch.
```

Slashes are not the only characters you can use to delimit strings. Most nonalphanumeric characters will work, as long as the delimiters are matched and do not occur in either string. For example, the following command substitutes the string A/3 for A/2 in the current line, using dollar signs ($) as delimiters:

```
*SUBSTITUTE$A/2$A/3$
    25          SIZE = A/3;
1 substitution
*
```

Note that a global substitution replaces all occurrences of the string, regardless of case, diacritical marks, or surrounding characters. If you want EDT to search for exact comparisons of case, use the SET SEARCH command and specify the parameter EXACT. (For information about the SET SEARCH command, see the *VAX EDT Reference Manual*.)

If the search string occurs in the middle of a longer string, the substitution will still be made. For instance, a global substitution of IN for AT would change all words containing the string AT. (LATER would become LINER, THAT would become THIN, SAT would become SIN, and so on.) To get EDT to prompt you before each substitution, use the /QUERY qualifier with the SUBSTITUTE command:

```
*SUBSTITUTE/ in / at /WHOLE/QUERY
    1       He was in the point of no return.
?y
    1       He was at the point of no return.
1 substitution
```

EDT prompts you for one of the following responses before each occurrence of the search string:

```
Y  Yes, do the substitution
N  No, do not do the substitution
Q  Quit, terminate the command
A  All, do the rest of the substitutions without query
```

1.3.6 Moving Text from One Location to Another

You can use the MOVE and COPY commands to move one or more lines of text from one place to another. The effect of these commands is similar; the only difference is that the COPY command does not delete the text from its original location, whereas the MOVE command does.

The syntax for the MOVE command is:

```
*MOVE first range TO second range/QUERY
```

When you use the MOVE command, EDT moves the lines in the first range above the line in the second range. EDT deletes the original copy of the text. For example, if you want to move lines 43 through 56 above line 13, enter the following command:

```
*MOVE 43 THRU 56 TO 13 RET
```

In the following example, EDT moves the current line above line 65.

```
*MOVE TO 65 RET
```

The second range always refers to a single line. EDT inserts the line or lines specified by the first range just above the line specified by the second range. The lines that are moved are renumbered to be consistent with their new location.

You can use the /QUERY qualifier with the MOVE and COPY commands to verify each line to be inserted in the range. (See Section 1.3 for more information about the /QUERY qualifier.)

Use the COPY command to copy text without deleting the original text.

```
*COPY first-range TO second-range/QUERY/DUPLICATE:n
```

When you use the COPY command, EDT copies the lines in the first range above the line in the second range. For example, if you want to copy lines 16 through 24 above line 3, enter the following command:

```
*COPY 16 THRU 24 TO 3
```

In the following example, EDT copies lines 15 through 60 to the position just before line 95.

```
*COPY 15 THRU 60 TO 95
```

You can use the /DUPLICATE qualifier with the COPY command if you want to insert the range of text more than once.

In the following example, EDT copies the first line and places it just above line 65. This operation is repeated 9 times.

```
*COPY BEGIN TO 65/DUPLICATE:9 RET
1 line copied 9 times
*
```

1.3.7 Replacing Text

The REPLACE command combines the DELETE and INSERT functions in one command. You can use REPLACE when you need to delete a block of text and want to type new text in that same location. The syntax for the REPLACE command is:

```
*REPLACE range [RET] text [CTRL/Z]
```

When you use the REPLACE command, EDT deletes the line or lines specified by the range. As with the DELETE command, EDT prints a message telling you how many lines you have removed.

If you supply no specifiers with the command, EDT deletes the current line. As soon as EDT finishes deleting the specified line or lines, it shifts to the insert state. (The cursor moves to the right, just as it does when you give the INSERT command.)

Anything you type after pressing RETURN is inserted. When you finish typing the new text, press CTRL/Z to signal EDT that you want to return to the asterisk prompt (*).

The following REPLACE command deletes the current line and shifts to the insert state:

```
*TYPE .
   17      This is the current line.
*REPLACE
1 line deleted
           I have just deleted the current line and
           am adding new lines to my text. <CTRL/Z>
   18      This is the next line.
*TYPE 16 THRU 18
   16      This is the line before the current line.
   16.1    I have just deleted the current line and
   16.2    am adding new lines to my text.
   18      This is the next line.
```

1.4 How to Use Nokeypad Mode

You can use nokeypad editing on VT200 Series, VT100, and VT52 terminals. You can use nokeypad commands to define keys.

The commands you use in nokeypad editing are English words or abbreviations that you enter from the main keyboard to move the cursor and edit text. As you type nokeypad commands, they are displayed on the lower left portion of the screen. As in line mode, you press RETURN to process the commands. You can join several commands on a single line and

process all of them by pressing RETURN. Note the following characteristics of the cursor in nokeypad mode:

- You will see the cursor at the top of the screen when you enter nokeypad mode.
- You can move the cursor by using the arrow keys.
- When you do not specify a range with your nokeypad command, you can affect the text at the cursor.

When you type two commands on the same line, such as 3W D-C, you have the option of separating them with a space (for the sake of clarity), even though EDT does not require one.

Nokeypad commands can be used to define keys. (See Section 1.9 for more information.)

1.4.1 Inserting Text

To insert text, enter the I command (insert command) and press RETURN. Whatever you type is inserted to the left of the cursor. When you want to start a new line of text, press RETURN to move the cursor to the next line. The following example demonstrates how to enter insert mode and type three lines of text.

```
I RET
This is the first line you type after entering insert mode. RET
You can type as many lines as you want. RET
This is the third. RET
```

If you want to correct a mistake or change a word, use the DELETE key to delete characters to the left of the cursor. If your mistake was on the previous line, you can continue to press DELETE until you reach the text you want to change. Using the DELETE key is the only way you can edit text while still in insert mode.

When you finish inserting the new text, press CTRL/Z to exit from insert mode. The I command disappears from the command line and you are ready to issue more nokeypad commands.

After you press CTRL/Z, you can use the arrow keys to move the cursor. Then you can use the I command again to insert new text at the cursor's new location.

```
I RET
On the first day of Christmas my true love gave to me, RET
a partridge in a pear tree. RET
CTRL/Z
```

If you need to insert less than a line of text, type the I command and follow it immediately by the text you want to insert. (If you put a space between the I and the text, EDT inserts a space in your text.) Press CTRL/Z immediately after the text and then press RETURN.

For instance, if you want to insert the word "Saturday" before the word "December" in the following example, move the cursor to the letter "D" in "December," type the letter I, type the word "Saturday, " (include the comma and the space), press CTRL/Z, and then press RETURN:

```
on December 13, 1985
ISaturday, CTRL/Z RET
on Saturday, December 13, 1985
```

1.4.2 Moving the Cursor

You can move the cursor by characters, words, or lines by specifying the entity. There are 22 different entities available in nokeypad mode. All the available entities are listed and described in the following table.

Entity	Description
C	Character—a single character. Characters include: letters, digits, punctuation marks, and control characters.
W	Word—a string of characters bounded by a predefined set of delimiters. The default delimiters are: space, horizontal tab, line feed, vertical tab, form feed, and line terminator.
L	Line—a single line of text starting just after a line terminator and encompassing all characters up to and including the next line terminator.
SEN	Sentence—a string of characters bounded by a predefined set of delimiters. The default delimiters for sentences are the period (.), exclamation point (!), and question mark (?), each of which must be followed by at least one space.
PAR	Paragraph—a string of characters bounded by a predefined delimiter. The default delimiter is two line terminators in succession.
PAGE	Page—a string of characters bounded by a predefined limit. The default delimiter is the form feed.

Entity	Description
SR	Select range—the string of characters between the position marked by the SEL command and the next cursor position.
BW BL BS BPAR BPAGE BR	Beginning of word, line, sentence, paragraph, page, buffer—the string of characters starting at the cursor and extending to the left until the beginning of the word, line, sentence, paragraph, page, or current buffer is reached.
EW EL ES EPAR EPAGE ER	End of word, line, sentence, paragraph, page, buffer—the string of characters starting at the cursor and extending to the right until the end of the word, line, sentence, paragraph, page, or current buffer is reached.
NL	Next line—the string of characters from the cursor to the beginning of the next line. The ending line terminator is included in the string.
"string"	String—all the characters between the previous cursor position and "string." The string must be enclosed with either single (') or double (") quotation marks. The string characters themselves are not affected by the command.
V	Vertical—the string of characters starting at the cursor and moving to the identical column position of the line above or below, depending on the specified direction. Vertical always includes a line terminator. If a line has fewer characters than the current vertical column, EDT moves to the last character in the line. If there is a horizontal tab, EDT moves left of the proper column to the tab character.

Use a count to move the cursor more than one entity at a time. The following examples demonstrate how to move the cursor using a count:

EXAMPLES

1 `3C`

This command moves the cursor forward three characters.

2 `4W`

This command moves the cursor forward four words.

3 5L

This command moves the cursor forward five lines.

If you want to set the direction of your operations backward, enter the BACK command. For example, after specifying the BACK command, the commands in the following examples are reversed.

EXAMPLES

1 3C

This command moves the cursor backward three characters.

2 4W

This command moves the cursor backward four words.

3 5L

This command moves the cursor backward five lines.

To set the direction of your operations forward again, enter the ADV command.

Use the plus (+) and minus (-) signs to override the ADV and BACK commands. The plus and minus signs only affect the commands they precede. For example, if you enter the command -3L, the cursor will move backward three lines. If you specify the BACK command and enter the command +2W followed by the command 3W, the cursor will move forward by two words, then backward by three words.

You can also use the four arrow keys to move the cursor.

The following table lists a sampling of nokeypad commands and the destination of the cursor after each command is entered.

Command	Destination of cursor
6C	forward six characters
-3W	backward three words
2L	forward two lines
-7C	backward seven characters
BACK 6C	backward six characters
BACK 4W	backward four words
+3SEN	forward three sentences
ADV PAR	forward one paragraph

Command	Destination of cursor
up arrow	up one character
down arrow	down one character
right arrow	right one character
left arrow	left one character

1.4.3 Locating Text

To locate text, you enter a series of characters (called a string) in quotation marks. EDT does not distinguish uppercase from lowercase or whether or not characters have diacritical marks in string searches unless you specify EXACT with the SET SEARCH command. (For information about the SET SEARCH command, see the *VAX EDT Reference Manual*.)

To set the direction of the search, use the ADV and BACK commands or precede the quote with a plus (+) or minus sign (-).

Type I to insert text, press RETURN, and enter the following text:

```
One little, two little, three little chickadees, four little,
five little, six little chickadees, seven little, eight little,
nine little chickadees, ten little chickadees feeding.
```

Press CTRL/Z to leave insert mode, enter the BACK command to set the direction backward, and press RETURN. Now search for the word "little" by entering "little" and pressing RETURN. (You must enclose the word with quotation marks.) The cursor will move to the last "little" you entered. To move the cursor to the "little" after the word seven, enter the following command line and press RETURN:

```
3"little"
```

To move the cursor to the first "little," enter the following command line and press RETURN:

```
6"little"
```

1.4.4 Deleting Text

To delete text, enter the letter D (the delete command) followed by an entity, as in DW (delete word) or DSEN (delete sentence). If the cursor is anywhere within a word and you type DW, the entire word is deleted. If the cursor is anywhere within a line and you type DL, the entire line is deleted. When you use D with B or E entities (for example, BSEN, EPAR), EDT deletes all the characters between the cursor and the beginning or end of the entity.

The examples that follow demonstrate various delete operations.

EXAMPLES

1 DC

This command deletes one character.

2 DL

This command deletes one line.

3 D3C

This command deletes three characters.

4 D4W

This command deletes four words.

5 DNL

This command deletes from the cursor to the beginning of the next line.

1.4.5 Undeleting Text

When you use the D command to delete characters, words, or lines, EDT stores the deleted entities in special buffers. Only the most recently deleted entity is stored. The last character you delete is stored in the delete character buffer; the last word you delete is stored in the delete word buffer; and the last line in the delete line buffer.

You can use undelete commands to insert the contents of these buffers into your text at any time. The three commands are:

countUNDC
countUNDW
countUNDL

UNDC inserts the contents of the delete character buffer, UNDW the delete word buffer, and UNDL the delete line buffer. Each command takes the count specifier, enabling you to repeat the insert in the same location.

You can use UNDC to draw a line. First, use the I command to insert one underscore in your buffer. Press CTRL/Z and RETURN to leave insert mode. Press the left arrow key (←) once to move the cursor to the underscore. Next, use the DC command to delete the underscore, putting it in the delete character buffer. Press RETURN. Then, use the UNDC command with a count of 50 to create your line:

```
I_ CTRL/Z RET
←
DC RET
50UNDC RET
```

You can use UNDW to create a line of "frogs." Enter the following commands:

```
Ifrog CTRL/Z RET
←
DW RET
25UNDW RET
```

1.4.6 Moving Text from One Location to Another

Moving and copying text is similar in both nokeypad and keypad mode. Nokeypad has three commands for copying and moving text: CUT, PASTE, and APPEND. Follow the three steps listed below to move text:

1. Use the CUT command to delete the text from its current location.
2. Move the cursor to the location where you want the text.
3. Use the PASTE command to insert the text at the new location.

To copy text, add one step:

1. Use the CUT command to delete the text from its current location.
2. Use the PASTE command to restore the text to its original location.
3. Move the cursor to the location where you want the text.
4. Use the PASTE command to insert the text at the new location.

1.4.7 Substituting Text

You can use the S (substitute) command to replace one string with another.

The syntax for the command is:

```
+ or - countS/old-string/new-string/
```

EDT searches the buffer in the current or specified direction until it finds the next occurrence of the old string. Then it deletes the old string and replaces it with the new string.

You can use count to perform as many substitutions as you need. Unlike the line mode substitution commands, S does not allow you to use a word like ALL or WHOLE to perform substitutions throughout the entire buffer. If you want to perform substitutions on every string, use a large number for count. When EDT cannot find another old string, it prints the message "String was not found" and stops. EDT has made all the substitutions up to that point.

EXAMPLES

1 `S/red/blue/`

EDT searches in the forward direction for the word "red" and replaces it with the word "blue."

2 `-S/good/bad/`

EDT searches in the backward direction for the word "good" and replaces it with the word "bad."

3 `12S/taste/flavor/`

EDT searches in the forward direction for the word "taste" and replaces it with the word "flavor." This operation is done twelve times.

4 `-4S/brown/grizzly/`

EDT searches in the backward direction for the word "brown" and replaces it with the word "grizzly." This operation is repeated three more times, in a backward direction.

You can use the plus (+) or minus sign (-) to control the direction of the search. The sign overrides EDT's current direction.

You can use any nonalphanumeric character as a delimiter, as long as the character is not used in one of the strings. For example, you could not use the slash as a delimiter to substitute the string "input/output" with "I/O." However, you could use another special character, such as:

```
S&input/output&I/O&
```

You can use the SN (substitute next) command only after you have established both the search string (old string) and the substitute string (new string). The syntax is:

```
+ or - countSN
```

You can specify a sign or count with SN. Use a sign to control the direction of the search and a count to indicate a certain number of substitutions.

To see how the SN command works, enter the following text:

```
October 13, 1985
October 19, 1985
October 24, 1985
October 30, 1985
```

Now, to change each occurrence of the word "October" to "December," enter the following commands:

```
-S/October/December/ RET
-3SN RET
```

1.5 Modifying Your EDT Environment

EDT provides many tools for modifying your own EDT environment. You can change the appearance of your screen display, the behavior of lines of text as you insert them, and the mode in which you are working. The following section discusses some of the SET commands provided by EDT. See the *VAX EDT Reference Manual* for descriptions of all the available SET commands.

1.5.1 Using SET Commands

You can use the SET commands to change the way EDT works. Some SET commands affect the display of text on your screen. For example, you can specify the number of lines you want displayed and whether or not the lines have line numbers (in line mode). You can also specify that EDT make an exact search. See the following SET commands:

```
SET LINES number
SET [NO]NUMBERS
SET SEARCH EXACT
```

By default, your screen contains 22 lines. If you want to decrease the screen display to 5 lines, enter the command SET LINES 5. If you are editing at slow data rates, you can increase your editing speed by decreasing the number of lines displayed on your screen.

When you are working in line mode, EDT displays line numbers by default. If you do not want the numbers to appear on the screen or paper, enter the command SET NONUMBERS.

By default, when you press the GOLD key followed by the FIND key, EDT searches for the specified string, disregarding the case of letters. For example, if you enter the search string "course," EDT will find every occurrence of the word (for example, "course," "Course," "COURSE"). But, when you enter SET SEARCH EXACT, EDT will only find occurrences of the word that exactly match the specified string ("course").

You can use the following line mode commands to determine which mode you enter:

```
SET KEYPAD
SET NOKEYPAD
```

In a startup command file, you would use the following SET commands to determine which mode you enter:

```
SET MODE LINE
SET MODE CHANGE
```

For example, to enter nokeypad mode from keypad mode, enter the following command after pressing GOLD/COMMAND:

```
Command: SET NOKEYPAD
```

One SET command, SET QUIET, even controls the sound of your terminal. You can use this command to suppress the terminal bell that signals an error.

1.5.2 Using SHOW Commands to See What Is Set

EDT provides the SHOW commands to enable you to see what is set and what is not set. For every SET command there is a SHOW command. If you want to see the number of lines on your screen, enter the SHOW LINES command. If you want to see whether EDT is performing an exact search, enter the SHOW SEARCH command.

For example, if you want to check the number of lines displayed on your screen, enter the following command:

```
*SHOW LINES
22
```

Or, if you want to see whether line numbers will be displayed in line mode, enter the following command:

```
*SHOW NUMBERS
numbers
```

The following table lists the SET commands discussed so far with their corresponding SHOW commands:

SET Command	SHOW Command
SET KEYPAD SET NOKEYPAD	SHOW KEYPAD
SET LINES	SHOW LINES
SET MODE LINE SET MODE CHANGE	SHOW MODE

SET Command	SHOW Command
SET NUMBERS SET NONUMBERS	SHOW NUMBERS
SET QUIET SET NOQUIET	SHOW QUIET
SET SCREEN	SHOW SCREEN
SET TRUNCATE SET NOTRUNCATE	SHOW TRUNCATE
SET WRAP SET NOWRAP	SHOW WRAP

For a complete list of the SHOW commands provided by EDT see the *VAX EDT Reference Manual*.

1.6 What Are Buffers?

Buffers are temporary holding areas for text. Buffers enable you to work with more than one file at a time. You can use the buffers available in EDT to:

- Divide one or more files into sections
- Move part or all of another file into your editing session
- Create a file from part or all of the text in a buffer

When you start an editing session, EDT automatically provides a buffer called MAIN. The MAIN buffer will serve as the work area for text you insert and edit. If you are editing a file that already exists, EDT puts a copy of the contents of the file into this MAIN buffer.

The MAIN buffer has the following characteristics:

- MAIN is automatically provided when you invoke EDT.
- MAIN exists during your entire editing session.
- MAIN cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

The other buffer that EDT provides automatically is called the PASTE buffer. When you use the CUT keypad function, the deleted text goes into the PASTE buffer. Every time you perform a new CUT operation, EDT clears the PASTE buffer and replaces its contents with the newly deleted text.

The PASTE buffer has the following characteristics:

- The PASTE buffer is automatically provided when you begin your editing session.
- The PASTE buffer can be edited.
- The PASTE buffer can be loaded with the contents of another file. (Use the INCLUDE command.)
- The PASTE buffer cannot be deleted. (Only the contents, not the name, of this buffer can be deleted.)

1.6.1 How to See Existing Buffers

If you want to see a list of the buffers in your editing session, enter the line mode command SHOW BUFFER. If you are working in your MAIN buffer and you have not used the CUT command or created any new buffers, you will see the following display:

```
=MAIN    23      lines
 PASTE   no      lines
```

(The number of lines used in this example, 23, is arbitrary.)

These two buffers, MAIN and PASTE, will always be displayed when you enter the SHOW BUFFER command because they always exist and you cannot delete them.

If you have just used the CUT and PASTE commands to move three lines of text and you enter the SHOW BUFFER command again, the display will indicate those three lines in the PASTE buffer:

```
=MAIN    20      lines
 PASTE   3       lines
```

The current buffer (the buffer in which you are working) is preceded by an equal sign (=). In the example above, MAIN is the current buffer. If you cannot remember which buffer you are working in, enter the SHOW BUFFER command.

Sometimes you will notice an asterisk (*) following the line count for the MAIN buffer. This asterisk indicates that EDT has not had a chance to read through the entire input file and has only seen as many lines as are indicated.

1.6.2 How to Create Buffers

To create a buffer from keypad mode, press the GOLD key, followed by the COMMAND function. When EDT prompts you with "Command:", enter the FIND command, then an equal sign (=) followed immediately by the buffer name of your choice. A buffer name can contain any alphanumeric characters, but it must begin with a letter. The only punctuation character you can use in a buffer name is an underscore (_).

To create a buffer named OSCAR, enter the following line after the command prompt:

```
Command: FIND=OSCAR
```

When you enter this command, the screen clears except for the [EOB] symbol, indicating that the current buffer, OSCAR, is empty. After you press RETURN, you can insert and edit text just as you would in the MAIN buffer.

To create a buffer from line mode, enter the FIND command, an equal sign (=) and the buffer name after the asterisk prompt (*). For example:

```
*FIND=OSCAR
```

To return to the MAIN buffer, type FIND=MAIN. The buffer named OSCAR will remain intact until you exit from the file, when only the MAIN buffer is saved.

You can also create a buffer during a MOVE operation. For example, to move lines 1 through 17 to a new buffer named MATKA, enter the following line mode command:

```
*MOVE 1:17 TO =MATKA
```

1.6.3 How to Delete Buffers

Use the line mode command CLEAR to delete buffers during an editing session. For example, to delete a buffer named CLAUDE, type the following command:

```
*CLEAR CLAUDE
```

You cannot delete the buffers MAIN and PASTE, only their contents. If you enter the command CLEAR MAIN, the contents of your MAIN buffer will disappear, but the buffer name will remain. When you exit from EDT, the contents of all the existing buffers, except MAIN, are deleted.

1.6.4 How to Copy Text from One Buffer to Another Buffer

To copy text from one buffer to another, enter the COPY command. For example, to copy the contents of a buffer named OSCAR to a buffer named YORICK, enter the following command:

```
*COPY =OSCAR TO =YORICK
```

When you complete this operation, the current buffer will be YORICK.

1.6.5 How to Copy Text from a File Into a Buffer

To copy text from a file, outside of EDT, into a buffer, use the INCLUDE command. For example, to copy the contents of a file named OUTSIDER.DAT to your MAIN buffer, type the following command:

```
*INCLUDE OUTSIDER.DAT =MAIN
```

1.6.6 How to Copy Text from a Buffer to a File

To copy text from a buffer to a file, use the WRITE command. You can use this command to copy text to a new file without affecting your editing session. For example, the following command puts a copy of lines 23 through the end of the current buffer into a file that you name EMT.DAT.

```
*WRITE EMT.DAT 23:END
```

The table below lists the three commands you need to copy text between buffers and external files.

Direction	Command
one buffer to another	* COPY =other_buffer TO =buffer
external file to buffer	* INCLUDE alien_file.typ =buffer
buffer to external file	* WRITE alien_file.typ n:n

1.7 Recovering from a Lost Editing Session

While you are editing or inserting text, EDT is keeping track of every keystroke you enter at your terminal. EDT records this information in a file called a journal file. Unless you specify otherwise, this journal file is deleted as soon as you give the EXIT or QUIT command. However, when you experience a system interruption, the journal file is not deleted.

The journal file does not contain a version of your text. Rather, it contains a record of the keystrokes you entered during the session. By combining the journal file with the text that you had at the beginning of your session, you can recover your session to a point just before the interruption.

Note

Sometimes, the last few keystrokes are missing. This is normal. No work from earlier in your session will be omitted.

The following example demonstrates how to recover a file named BIZARRE.RNO after the editing session has been interrupted.

```
$ EDIT/RECOVER BIZARRE.RNO
```

When you work with journal files, you will notice that they have a file type of JOU. The file name is the same as the file you were editing. The journal file for BIZARRE.RNO is BIZARRE.JOU. And, the journal file for HAMMER.LIS is HAMMER.JOU. When you enter the EDIT/RECOVER command, you enter the name of the file with its original file type, not the JOU file type. Otherwise you will edit the journal file.

To recover an editing session you had begun with the command EDIT PLIERS.DAT, enter the following command line:

```
$ EDIT/RECOVER PLIERS.DAT
```

Notice that you enter the name of the file you were editing, not the name of the journal file.

After you enter the EDIT/RECOVER command, EDT replays the editing session. When it finishes, you can continue editing.

You can start an editing session and interrupt it to see how the /RECOVER qualifier works by following the steps listed below:

1. Invoke EDT to create a file named FUN.FUN.
2. Insert text into FUN.FUN and perform a number of edits.
3. Press CTRL/Y to end the editing session abruptly. (You will be at the DCL command level at this point.)

4 Enter the DCL command DIRECTORY to see the newly created journal file (FUN.JOU).

5 Invoke EDT again using the /RECOVER qualifier to recover your lost session by entering the command line:

```
$ EDIT/RECOVER FUN.FUN
```

6 Ensure that your last few keystrokes were recovered and continue to edit the file. Then enter the EXIT or QUIT command to terminate your editing session.

7 Enter the DCL command DIRECTORY again. The journal file, FUN.JOU, will not be there because you have exited normally from EDT.

1.8 How to Create Columns and Layered Text

You can use EDT's tabbing facility in keypad mode to format text in two ways: (1) columns of eight and (2) layers.

1.8.1 Creating Columns of Eight

EDT has automatic tabs set for columns eight characters wide. Therefore, when you press TAB, your text will move to the nearest preset tab stop: column 9, 17, 15, 33, 41, 49 and so on. TAB will always move your text to the right regardless of EDT's current direction. If you want to move text back toward the left margin, use the DELETE key.

The following example demonstrates how to use TAB. Enter the following lines of text:

```
Tab each word in this sentence?
Tab each word in this sentence?
Tab each word in this sentence?
```

Move the cursor to the first letter of the first word and press TAB. Then move the cursor to the first letter of the second word and press TAB; then, to the third word and press TAB and so on. When you are done, the sentences will look like this:

```
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
Tab     each    word    in      this    sentence?
```

The words will be organized in columns of eight characters. As long as you want to arrange text in columns of eight, you can use TAB.

1.8.2 Creating Layers of Text

If you have layered text, like an outline, you can use the line mode command SET TAB with the TAB key to format the lines. When you enter the SET TAB command, and specify a value, for example 15, you can tab your first item over 15 spaces by pressing TAB. The following example shows a list of colors:

```
red
green
beige
blue
white
```

After you enter the following line mode command, move the cursor to the first letter of each word in your list of colors and press TAB.

```
*SET TAB 15
```

Your tabbed list is shown below:

```
        red
        green
        beige
        blue
        white
```

The two most important things to remember when you are using SET TAB are:

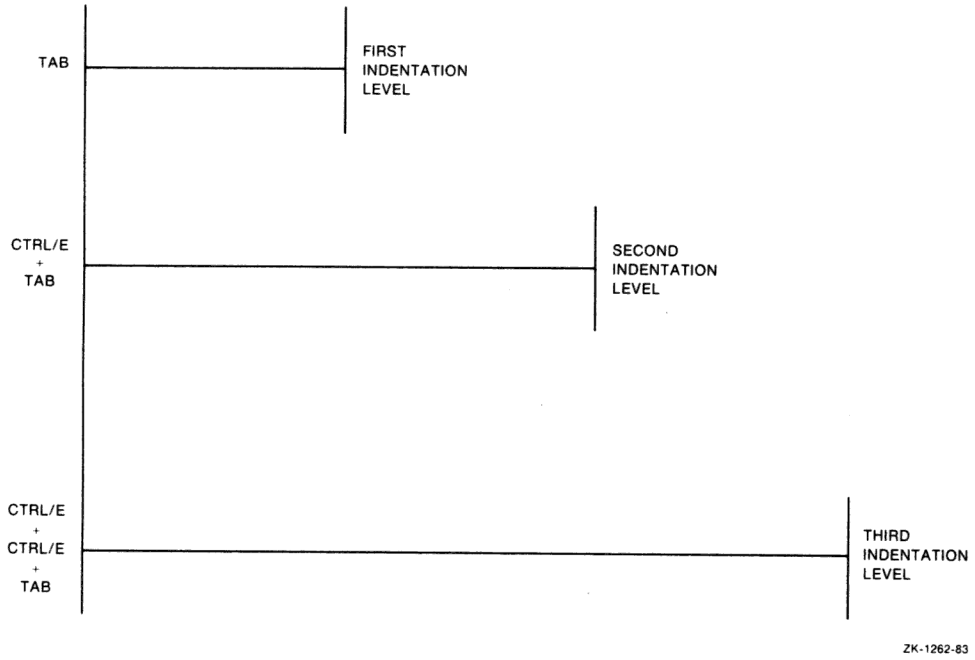
- The SET TAB value only affects the first indentation on a line.
- The cursor must be at the left edge of the screen on the line being moved.

To format an outline, use the line mode command SET TAB with the keys CTRL/E, CTRL/D, and TAB to change the indentation level of your lines.

Use CTRL/E to increase the indentation level. After you enter the SET TAB command, you can move the first line to be indented over to the first indentation level by pressing TAB. The first indentation level is the same as the SET TAB value, but, when you want to move a line to the second indentation level, you need to use CTRL/E.

Figure 1–5 shows the keys, TAB and CTRL/E, with their corresponding indentation levels.

Figure 1–5 Using CTRL/E to Increase the Indentation Level



Each time you need to increase the indentation level, press CTRL/E. Then, press TAB to actually move the line of text.

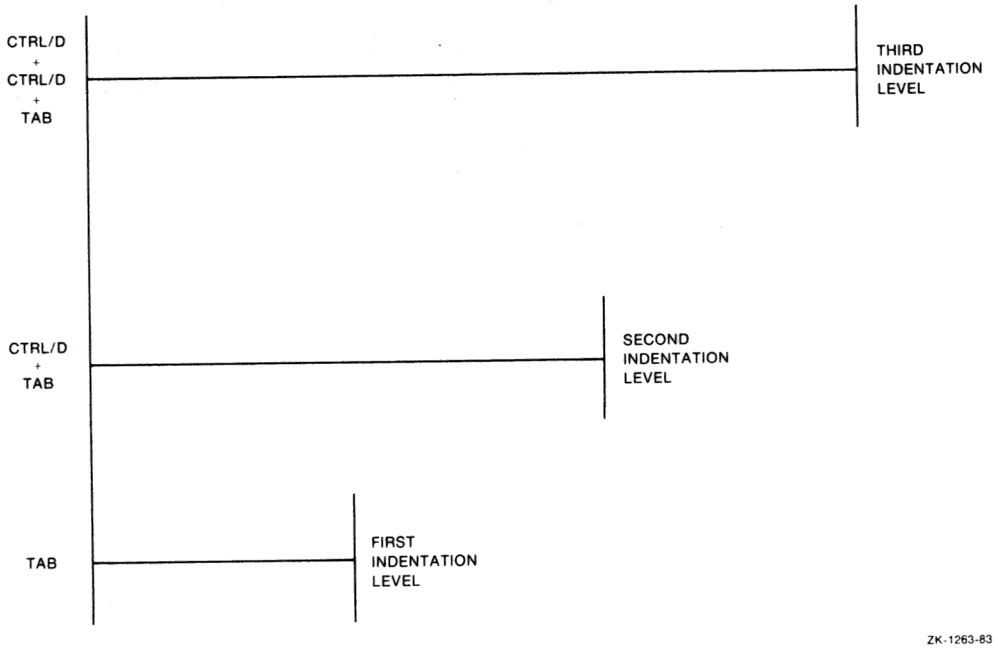
For example, enter the command SET TAB 10, press TAB, and type the word TEN. Press RETURN. Then press CTRL/E, followed by TAB, and type the word TWENTY. Press RETURN. Next, press CTRL/E again, TAB, and type the word THIRTY. Press RETURN. Finally, press CTRL/E, TAB, and type the word FORTY. Your display will look like this:

```
TEN
        TWENTY
                THIRTY
                        FORTY
```

Use CTRL/D to decrease the indentation level. CRTL/D does the opposite of CTRL/E. It lowers the indentation level count by one. It does not move text back toward the left margin. Rather, it lessens the amount that text is moved to the right.

Figure 1–6 shows the keys, TAB and CTRL/D, with their corresponding indentation levels.

Figure 1–6 Using CTRL/D to Decrease the Indentation Level



The following sampling shows part of an outline before you indent the lines:

```
I.
A.
B.
1.
C.
II.
```

After you perform the steps listed below, the outline will look like this:

```
I.
    A.
    B.
        1.
    C.
II.
```

1. Choose the number of spaces you want to indent (for example, 5).
2. Enter the SET TAB command followed by that number.
3. Move the cursor to the roman numeral I and press TAB.

4 Move the cursor to the letter A, press CTRL/E, and press TAB.

5 Move the cursor to the letter B and press TAB.

6 Move the cursor to the number 1, press CTRL/E, and press TAB.

7 Move the cursor to the letter C, press CTRL/D, and press TAB.

8 Move the cursor to the roman numeral II, press CTRL/D, and press TAB.

An outline and the tabbing keys you press to create the outline are shown below:

```
                Malcolm's Family Tree
TAB
                    I. Malcolm
CTRL/E + TAB
                       A. Ian
TAB
                       B. Tatiana
TAB
                       C. Lars
CTRL/E + TAB
                                        1. Stephanie
TAB
                                        2. Brian
CTRL/D + CTRL/D + TAB
                    II.Frederick
CTRL/E + TAB
                       A. Louis
CTRL/E + TAB
                                        1. Elsa
CTRL/D + CTRL/D + TAB
                    III.Louisa
CTRL/E + TAB
                       A. Jeffrey
TAB
                       B. Andrew
CTRL/D + TAB
                    IV.Sonya
CTRL/E + TAB
                       A. Beth
TAB
                       B. Celia
TAB
                       C. Franklin
CTRL/E + TAB
                                        1. Martin
CTRL/E + TAB
                                           a) Stephan
CTRL/D + TAB
                       D. Martha
```

Note

There are only two keypad tabbing functions that actually move text: TAB and CTRL/T. The remaining three functions (CTRL/A, CTRL/D, and CTRL/E) determine the effect that the first two have on your text. CTRL/A and CTRL/T are discussed in the following sections.

1.8.3 Using CTRL/A to Indent Text

You can use CTRL/A to move a line of text to an indentation level greater than 1, without having to enter repeated tab increments (CTRL/E). There are three steps to the process:

1. Issue a SET TAB command, specifying a value.
2. Move the cursor to one of the indentation levels. The indentation level must be a character position that is a multiple of the value you specified with SET TAB. For example, you could move the cursor to 20 or 25 if the tab size is 5.
3. Press CTRL/A.

Once you choose the indentation level you want, and establish the position with CTRL/A, you can use TAB to indent the line of text to that place.

If you want to indent text to a different indentation level after establishing a position with CTRL/A, you can either use CTRL/A again to reset the position or use CTRL/E and CTRL/D to adjust the indentation level up or down.

Note

If the cursor position is not evenly divisible by the SET TAB value when you press CTRL/A, EDT displays the message "Could not align tabs with cursor."

1.8.4 Using CTRL/T to Indent Groups of Lines of Text

Use CTRL/T to indent groups of lines using the current SET TAB value. The CTRL/T function is similar to CTRL/A, CTRL/D, and CTRL/E in the following two ways:

1. CTRL/T has no effect unless you have specified a value with the SET TAB command.
2. CTRL/T moves entire lines of text.

On the other hand, CTRL/T differs from these other CTRL tabbing functions in the following two ways:

1. When you press CTRL/T, you do not press TAB to make the block of text move to the right. The text moves automatically.
2. The block of text is indented only the amount of the value established with the SET TAB command, without regard to the current indentation level.

To allow CTRL/T to work correctly in EDT, you must disable DCL control over CTRL/T. By default, DCL displays process statistics when you press CTRL/T. To disable DCL control, enter the following command at the DCL command level:

```
$ SET NOCONTROL=T
```

To indent two or more lines of text using CTRL/T, create a select range of those lines, and then press CTRL/T. EDT indents the range of lines only to the current SET TAB value. To indent text by multiples of the SET TAB value, use the GOLD repeat function.

Note

All the functions, except for TAB, require that a SET TAB value be established for your editing session. EDT's default is SET NOTAB. As long as NOTAB is in effect, pressing CTRL/A, CTRL/D, CTRL/E, or CTRL/T has no effect on the text you are editing.

1.8.5 Looking at the Indentation Level

Use the SHOW TAB command to see the value specified with the SET TAB command and the current indentation level. For example, if you set your tab size to 15 and press CTRL/E twice, EDT will display the following information when you issue the SHOW TAB command:

```
*SHOW TAB
tab size 15; tab level 3
```

For more information on the SHOW TAB command, see the *VAX EDT Reference Manual*.

1.9 Why Define Keys?

When you define or redefine a key, you change the function of that key. You can use combinations of nokeypad commands or existing keypad functions to dictate what a key will do.

Some advantages of defining keys are listed below.

- You can take advantage of different entities available in nokeypad mode. For example, you can define keys to delete sentences, paragraphs, and pages.
- You can combine several nokeypad commands into one key definition. For example, you can define a key to find a word, delete it, and substitute several words in its place.
- You can access nokeypad commands that do not exist in keypad editing, such as SHL, SHR, CHGL, CHGU, and DATE.

1.9.1 How to Define a Key

EDT provides two commands to define keys: CTRL/K and DEFINE KEY. The line mode command DEFINE KEY allows you to put key definitions in startup command files and EDT macros, as well as create key definitions during your EDT session. You can only use CTRL/K when working in keypad mode.

1.9.1.1 Using CTRL/K to Define a Key

Follow the steps below to define a key in keypad mode:

1. Press CTRL/K.
2. Press the key (or key sequence) you want to define (for example, GOLD /A or CTRL/H).
3. Type in the key definition. (You will either enter a string of nokeypad commands and/or press a sequence of keypad function keys.)
4. Type a period (.).
5. Press the ENTER key.

In order to define keys, you need to be familiar with nokeypad commands. See Section 1.9 and the *VAX EDT Reference Manual* for more information.

The following example uses D and SEN. D is the delete command and SEN refers to a string of characters (a sentence) enclosed by delimiters. This example defines CTRL/A to delete the sentence at your cursor.

```
CTRL/K
Press the key you wish to define
CTRL/A
Now enter the definition terminated by ENTER
DSEN.<ENTER>
```

The following example demonstrates how to define a key in keypad mode to draw a vertical line.

```
CTRL/K
Press the key you wish to define
CTRL/V
Now enter the definition terminated by ENTER
22(+VI|^Z).<ENTER>
```

In this example, CTRL/K tells EDT to define the key combination CTRL/V. The number 22 tells EDT to execute the commands in the parentheses 22 times. The definition for the down arrow is +V. The command I| tells EDT to put the vertical bar character into your text. The insert command is terminated by ^Z. The period (.) ensures that the command will take effect as soon as you press the key.

Note

The ^Z is typed as a circumflex followed by the letter Z.

1.9.1.2 Using the DEFINE KEY Command

Use the following syntax with the DEFINE KEY command:

```
*DEFINE KEY key name AS "string."
```

Key name is the name of the key or its keypad number, for example, DEFINE KEY GOLD F or DEFINE KEY 3. (See Figure 2-1 for keypad diagrams showing key names and keypad numbers. Note that you use the small number in the lower right corner on the keypad diagram. For example, you would use the number 11 to redefine keypad key PF3.) Type the *string*, which is the actual definition, completely in nokeypad syntax. Follow the nokeypad commands with a period (.), and enclose the definition (and the period) in quotation marks (") or apostrophes (').

Note

To use a key you have defined with the DEFINE KEY command, you must be in keypad mode.

The steps for defining a key using the DEFINE KEY command are outlined below:

1. Type the DEFINE KEY command.
2. Type the key name you want to define followed by the word *AS* (for example, DEFINE KEY CONTROL B AS . . .)
3. Enter a string of nokeypad commands followed by a period (.). Enclose the string and the period in quotation marks (") (for example, DEFINE KEY CONTROL B AS "D+NL.")
4. Press RETURN.

In the following example, the keypad key 5 is redefined to insert the line of text "Fill in all the blanks. Please print.":

```
*DEFINE KEY 5 AS "IFill in all the blanks.  Please print.^Z."
```

In this example, the command to insert text is enclosed in quotation marks. The following components are in the quotation marks:

- The INSERT command (I)
- Text to be inserted
- ^Z (to complete the insertion). Remember to enter the ^Z by typing the circumflex followed by the letter Z.
- A period (.) to enable the command to take effect as soon as you press the key

When you use the DEFINE KEY command to define CTRL keys, for example, CTRL/A, type the word CONTROL:

```
*DEFINE KEY CONTROL A AS "string."
```

When you want to define the GOLD key, for example, GOLD 5, type the word GOLD:

```
*DEFINE KEY GOLD 5 AS "string."
```

When you want to redefine a function key on the LK201 keyboard, type the word FUNCTION. For example:

```
*DEFINE KEY FUNCTION 29 AS "string"
```

To create key definitions, you can consolidate several keypad functions into one key definition. The following table lists the key names you use when you define keys:

Key name	VT100	VT52	LK201
20	PF1	red key	PF1
10	PF2	blue key	PF2
11	PF3	black key	PF3
17	PF4	-	PF4
0-9	keypad 0-9	keypad 0-9	keypad 0-9
16	period key	period key	period key
19	comma key	comma key	comma key
18	minus key	minus key	minus key
21	ENTER key	ENTER key	ENTER key
FUNCTION 1	-	-	Find
FUNCTION 2	-	-	Insert Here
FUNCTION 3	-	-	Remove
FUNCTION 4	-	-	Select
FUNCTION 5	-	-	Previous Screen
FUNCTION 6	-	-	Next Screen
FUNCTION 28	-	-	Help
FUNCTION 29	-	-	Do

You can combine keypad functions to correct typing errors. For example, to correct reversed letters, you delete a character, press the right arrow key, and undelete the character. Suppose you type the word "social" as "oscial." You would move the cursor to the "o," delete it, press the right arrow key to move the cursor to the "c," and undelete the "o." The following table displays these steps translated into nokeypad commands:

Function	Nokeypad Commands
delete a character	D+C
move cursor one character to the right	+C
undelete a character	UNDC

To define CTRL/R to do all three commands at once (thus saving you keystrokes), enter the following line:

```
*DEFINE KEY CONTROL R AS "D+C +C UNDC."
```

Now, when you press CTRL/R, EDT will reverse the two characters at the cursor.

1.9.2 Which Keys Can Be Defined

You can define all keypad keys, all function keys, and GOLD with either a keypad key or a function key.

```
*DEFINE KEY 8 AS "string."
*DEFINE KEY GOLD 8 AS "string."
*DEFINE KEY FUNCTION 34 AS "string."
```

You can define GOLD/keyboard keys. A GOLD/keyboard key refers to the combination of GOLD with a key on the main keyboard.

```
*DEFINE KEY GOLD A AS "string."
```

When you define a GOLD/keyboard key sequence, enclose the following symbols with quotation marks ("!"):

exclamation point (!)
percent sign (%)
apostrophe (')
quotation marks (")

```
*DEFINE KEY GOLD "!" AS "string."
```

Note

When you define a quotation mark ("), surround it with apostrophes ('). For example, DEFINE KEY GOLD '"' as "string". And, when you define an apostrophe, surround it with quotation marks ("'").

You can also define keys using the control key (CTRL) with letter keys as well as with brackets ([ ]), a backslash (\), a tilde (~), a circumflex (^), and an underscore (_).

```
*DEFINE KEY CONTROL A AS "string."
*DEFINE KEY CONTROL ] AS "string."
```

You can also define GOLD CONTROL keys (for example, DEFINE KEY GOLD CONTROL AS "string"). When you use GOLD and CONTROL, press the GOLD key first, then hold down the CTRL key while you press the keyboard key.

Do not define the following key combinations:

CTRL/C
CTRL/O
CTRL/Q
CTRL/S
CTRL/T (unless you SET NOCONTROL=T)
CTRL/X
CTRL/Y

GOLD/CTRL/C
GOLD/CTRL/O
GOLD/CTRL/Q
GOLD/CTRL/S
GOLD/CTRL/X
GOLD/CTRL/Y
GOLD/digit (keyboard keys 0 through 9)
GOLD/minus sign (-)

You can redefine the following key combinations, but be aware of their default functions:

CTRL/H (BACKSPACE)
CTRL/I (TAB)
CTRL/J (LINEFEED)
CTRL/M (RETURN)

1.9.3 How to Save Defined Keys

Once you define a key, it stays defined throughout your editing session until you type EXIT or QUIT or redefine the key. You can save key definitions, however, by including them in your startup command file or a macro file. A sample section from a startup command file follows:

```
DEFINE KEY CONTROL R AS "EXT INCLUDE ?'INCLUDE FILE: '."❶
DEFINE KEY CONTROL G AS "EXT FIND=MAIN."❷
DEFINE KEY CONTROL L AS "EXT EXIT."❸
DEFINE KEY CONTROL N AS "EXT QUIT."❹
```

When you press:

❶ CTRL/R, EDT prompts you for a file to include.

❷ CTRL/G, EDT puts you at the top of the MAIN buffer.

❸ CTRL/L, EDT exits your editing session.

❹ CTRL/N, EDT quits your editing session.

See Section 1.11 for more information about saving key definitions in startup command files.

1.10 How to Use Macros

EDT allows you to define a series of line mode commands as a macro. The following sections explain how and why you would want to do this.

1.10.1 What Is a Macro?

A macro is a sequence of line mode commands that you can execute when you type the macro name. You use the DEFINE MACRO command to add the macro name to the list of valid line mode commands for the duration of your editing session.

1.10.2 How to Create a Macro

To create a macro, assign a name, for example CONCERTS, with the DEFINE MACRO command:

```
*DEFINE MACRO CONCERTS
```

Next, enter a buffer of the same name:

```
*FIND=CONCERTS
```

Then type the list of line mode commands, in this case use the INSERT command to create a list of dates. Type the INSERT command followed by the semicolon, one space, and then the text to be inserted. End the list with CTRL/Z to invoke the line mode prompt. (To return to the current line in the MAIN buffer, type the FIND command.)

```
*INSERT
            INSERT; September 20, 1985
            INSERT; November 13, 1985
            INSERT; December 13, 1985
            INSERT; January 19, 1986
            INSERT; March 15, 1986
            INSERT; May 5, 1986
            INSERT; December 14, 1987
^Z
[EOB]
*FIND=MAIN
```

EDT will execute that series of commands whenever you type the name of the macro as a line command. For example:

```
*CONCERTS
*TYPE WHOLE
    1       September 20, 1985
    2       November 13, 1985
    3       December 13, 1985
    4       January 19, 1986
    5       March 15, 1986
    6       May 5, 1986
    7       December 14, 1987
```

1.10.3 What Macros Can Do

Although macros can contain only line mode commands, they are able to perform a variety of functions.

- Macros can contain SET commands to tailor your editing session.
- Macros can contain a series of line mode commands.
- Macros can contain a series of key definitions.
- Macros can define other macros or call up macros from external files.

1.10.4 How Macros Are Like Startup Command Files

Both macros and startup command files contain line mode commands, but macros are more flexible. A startup command file (see Section 1.11) is executed once at the beginning of an editing session. But, you can create and access many different macros containing different sets of commands during one editing session just by typing the macro name.

1.10.5 The Life of a Macro

Macros can be created at any time during your editing session and can be used for the remainder of the session. Since they are located in buffers, they disappear as soon as you leave EDT. If you create a macro that you want to save for other EDT sessions, you can use the EDT command WRITE to put a copy of the macro in an external file. When you need that macro again, use the INCLUDE command to copy the macro into a buffer and then establish the macro name as a command with the DEFINE MACRO command.

The following example illustrates this process:

```
$ EDIT FUN.FUN
*DEFINE MACRO HEADING
*FIND=HEADING
*INSERT
INSERT;  Name:
INSERT;  Address:
INSERT;  Number:
INSERT;  Date:
^Z
[EOB]
*WRITE HEADING.MAC
DISK$:[HARTWELL]HEADING.MAC;1 4 lines
*EXIT
   .
   .
   .
$ EDIT WHY.NOT
*HEADING ❺
 ^
Unrecognized command
*FIND=HEADING
*INCLUDE HEADING.MAC ❻
*DEFINE MACRO HEADING ❼
*FIND=MAIN ❽
*HEADING ❾
*TYPE WHOLE

    Name:
    Address:
    Number:
    Date:
```

❶ The DEFINE MACRO command makes HEADING a valid line mode command.

❷ You enter a buffer named HEADING.

❸ You use four INSERT commands to create the text you want to use.

❹ The WRITE command puts a copy of the macro in an external file.

❺ In a new editing session, EDT does not recognize the macro named HEADING until you define the macro with the DEFINE MACRO command.

❻ The INCLUDE command brings the macro in from an outside file and puts it in the buffer named HEADING.

❼ The DEFINE MACRO command makes HEADING a valid line mode command.

❽ You return to the MAIN buffer. (You must leave the buffer containing the macro before you use the macro as a line mode command.)

❾ The HEADING command now functions as a line mode command.

Note

Macros override line mode commands. If you create a macro with the same name as an existing line mode command, EDT performs the macro, not the line mode command. As long as the macro is in effect, it overrides the default line mode command. If you need to reestablish the default use of the line mode command, use the CLEAR command to eliminate the buffer with the same name.

1.10.6 Including Specifiers in a Macro

When you use line mode commands that take specifiers, you must include the appropriate specifier in the macro text. For example, if you create a macro that includes a substitute command, you must supply the strings for EDT to use with that command.

You cannot use the SUBSTITUTE command in a macro and expect to enter the strings after typing the macro name.

The same rule applies when you use a command like TYPE in your macro. If you want to have a command that automatically types the current line and the next 9 lines, you can create the macro XTYPE to be:

```
TYPE . THRU +9
```

You cannot, however, create a command to give you a choice of how many lines you want to type allowing you to enter the final digit from the terminal.

1.11 What Is a Startup Command File?

A startup command file contains EDT line mode commands that are executed when you invoke EDT—before you receive control of the editor. You can use startup command files to customize your EDT sessions. You can create startup command files in either your current or main directory. Some of the line mode commands that a startup command file might contain are:

- DEFINE KEY commands—to redefine the function invoked by a function key, a keypad key or a control character while you are editing in keypad mode.
- DEFINE MACRO commands—to associate a name with a sequence of line editing commands stored in a text buffer. You can then invoke the sequence by entering the macro name in response to the line editing asterisk prompt.

- INCLUDE commands—to bring text from a file into a text buffer. You might use them to load macros into a buffer, or to fill a buffer with text that you often use.
- SET commands—to establish EDT operating parameters. For example, SET TAB establishes the increment for structured tabs, and SET MODE CHANGE invokes keypad mode.

The following example demonstrates how to create a two-line startup command file.

```
$ EDIT SETUP.EDT
Input file does not exist
[EOB]
*INSERT
        SET MODE CHANGE
        SET LINES 5
CTRL/Z
[EOB]
*EXIT
```

In this example, the name of your startup command file is SETUP.EDT. It contains two SET commands: SET MODE and SET LINES. SET MODE CHANGE causes EDT to begin your editing session in keypad mode and SET LINES 5 limits the display of text on your screen to five lines.

Use the qualifier /COMMAND= with the EDIT command to indicate the startup command file you want EDT to use. The following command line tells EDT to execute the line mode commands in the startup command file named SETUP.EDT to edit a file named FUN.FUN:

```
$ EDIT/COMMAND=SETUP.COM FUN.FUN
```

When you begin an editing session, EDT automatically searches your current directory to see if it contains a startup command file named EDTINI.EDT. By default, if EDT finds that file, it executes the commands the file contains before turning control over to you. Because EDTINI.EDT is the default startup command file, you do not need to name it in the command line. For example, if you create an EDTINI.EDT file and fill it with the following line mode commands, EDT will execute these commands the next time you invoke EDT:

```
$ EDIT EDTINI.EDT
Input file does not exist
[EOB]
*INSERT [RET]
        SET QUIET
        SET NONUMBERS
[CTRL/Z]
*EXIT
$ EDIT LIST.DAT
```

❶ EDT is invoked to create the file named EDTINI.EDT.

❷ SET QUIET suppresses the bell sound when EDT prints an error message (in keypad mode).

❸ SET NONUMBERS suppresses line numbers in line mode.

❹ Now, when you invoke EDT to edit a file named LIST.DAT, the commands in EDTINI.EDT will execute before you gain control of the editor.

The following startup command file contains two SET commands, three DEFINE KEY commands, and one DEFINE MACRO command:

```
SET WRAP 60 ❶
SET SEARCH EXACT ❷
DEFINE KEY GOLD W AS "CHGUW."❸
DEFINE KEY CONTROL B AS "EXT INCLUDE ?'INCLUDE FILE: '."❹
DEFINE KEY CONTROL G AS "EXT FIND=MAIN."❺
FIND=GENERAL❻
INSERT ;SET SEARCH GENERAL❼
DEFINE MACRO GENERAL❽
FIND=MAIN❾
```

❶ Limits the line length for inserting text in keypad mode to 60 characters.

❷ Causes EDT to match the case and diacritical marks (for example, grave accent or circumflex) of letters in search strings exactly.

❸ Defines GOLD/W to change all lowercase letters in a word to uppercase.

❹ Defines CTRL/B to extend the INCLUDE line command to keypad mode and issue a prompt.

❺ Defines CTRL/G to return to the MAIN buffer.

❻ Creates a buffer named GENERAL and moves there.

❼ Inserts the line mode command "SET SEARCH GENERAL."

❽ Adds "GENERAL" to the list of valid line mode commands.

❾ Returns to the first line of the MAIN buffer.

2 Editing Files With VAXTPU (EDT Keypad Emulator)

The EDT Keypad Emulator is an editing interface to the VAX Text Processing Utility (VAXTPU). It emulates the keypad commands and a subset of the line editing commands of the EDT editor, but does not provide nokeypad editing. The purpose of the EDT Keypad Emulator is to provide you with a familiar editing interface that you can extend with the powerful capabilities of VAXTPU. You can use the EDT Keypad Emulator to create new files, insert text into them, and edit and manipulate that text. You can also edit and manipulate text in existing files.

Because this chapter assumes that you are familiar with the EDT editor, it does not provide guidelines for using EDT. Instead, it outlines the differences between the EDT Keypad Emulator and EDT and provides guidelines for extending the EDT Keypad Emulator using VAXTPU. Chapter 1 contains guidelines for using EDT.

2.1 Invoking and Terminating the EDT Keypad Emulator

Invoke the EDT Keypad Emulator by typing the following DCL command:

```
$ EDIT/TPU/SECTION=EDTSECINI
```

Terminate an editing session with the EXIT or QUIT command. Like EDT, the EDT Keypad Emulator does not destroy the contents of any existing file that you edit; it simply produces a new version of a file when you terminate the EDT Keypad Emulator with the EXIT command.

2.1.1 Invoking the EDT Keypad Emulator

You may start an editing session by creating a new file and then insert text into that file during the session. You may name this file either when you invoke the EDT Keypad Emulator by specifying a file specification, or when you exit from the EDT Keypad Emulator by responding to the following prompt:

```
Enter a file name to write buffer MAIN: else press Return.
```

You may also specify an existing file when you invoke the EDT Keypad Emulator.

You must specify the /SECTION=EDTSECINI qualifier with the EDIT/TPU command to execute the EDT Keypad Emulator section file. You may specify other optional command qualifiers when you invoke the EDT Keypad Emulator. See Appendix G of the *VAX Text Processing Utility Reference Manual* for a complete description of the EDIT/TPU command.

To invoke the EDT Keypad Emulator to edit an existing file named THIS.DAT, enter the following command:

```
$ EDIT/TPU/SECTION=EDTSECINI THIS.DAT
```

If your current default directory contains a file named THIS.DAT, the contents of the latest version of THIS.DAT are displayed on your terminal screen. The cursor is positioned at the top of the screen and the EDT Keypad Emulator is ready to insert text.

```
When the madrigal group sang a loud and silly
song about John Brown, the man in the front row
sat up with a start.
[End of MAIN]
%TPU-S-FILEIN, 3 lines read from file WORKDISK:[USER]THIS.DAT
```

To create a new file named STUFF.DAT in your current default directory, enter the following command:

```
$ EDIT/TPU/SECTION=EDTSECINI STUFF.DAT
```

Only the [End of MAIN] symbol and the TPU and RMS messages appear on the screen. The EDT Keypad Emulator is ready to insert text.

```
[End of MAIN]
%TPU-E-OPENIN, error opening WORKDISK:[USER]STUFF.DAT; as input
-RMS-E-FNF, file not found
```

You may wish to use a command symbol to invoke the EDT Keypad Emulator. For example, if you place the following statement in your login command file and execute your login command file, you will only need to type EDTEM to invoke the EDT Keypad Emulator.

```
$ EDTEM == "EDIT/TPU/SECTION=EDTSECINI"
```

2.1.2 Terminating an EDT Keypad Emulator Session

Both the EXIT and QUIT commands terminate an editing session. The EXIT command saves your edits in a new version of the file. The QUIT command terminates the editing session without saving the edits. Any existing versions of the file remain unchanged regardless of how the editing session is terminated.

2.1.2.1 Saving Your Edits

The EXIT command terminates an editing session and writes the contents of the MAIN buffer to an output file. Type the EXIT command at the asterisk prompt (*). To enter the EXIT command, press CTRL/Z. The cursor moves to the asterisk prompt indicating that you can enter an EDT emulator command. Type EXIT and the contents of the main buffer are written to an output file. For example, if you use the EXIT command after editing a file named FUN.DAT;1, the output file is named FUN.DAT;2.

```
* EXIT
%TPU-S-FILEOUT, 54 lines written to file WORKDISK:[USER]FUN.DAT;2
$
```

2.1.2.2 Discarding Your Edits

To terminate the EDT Keypad Emulator without saving your edits, press CTRL/Z and enter QUIT at the asterisk prompt.

```
And the moral of the story is...
[End of MAIN]
*QUIT
Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press RETURN if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press RETURN, and exit from the file as described in the previous section.

2.1.3 Using the Help Facility

The EDT Keypad Emulator's help facility allows you to get help during your editing session without disturbing your work.

2.1.3.1 Getting Help on Keypad Editing Commands

Press the HELP key (or the PF2 key on the VT100) to obtain a list and diagram of keypad editing keys. Press the key for which you need help, and the help text for that key will appear on your screen.

2.1.3.2 Getting Help on EDT Keypad Emulator and VAXTPU Commands

To obtain help for EDT Keypad Emulator commands (VAXTPU equivalents for EDT line editing commands that the emulator does not support) and VAXTPU commands, press CTRL/Z and enter HELP at the asterisk prompt. When you have finished using the help facility, press CTRL/Z followed by CTRL/F to resume editing.

2.2 Recovering from Interruptions

You can recover from interruptions to your editing session in the following ways:

- Resuming an interrupted editing session—The DCL command CONTINUE resumes an EDT Keypad Emulator editing session that was interrupted by pressing CTRL/Y, as long as only built-in DCL commands were entered after CTRL/Y.
- Recovering from an interrupted editing session—By default, the EDT emulator keeps a journal file with the same file name as the input file and a file type of TJL. If an editing session ends without interruption, the journal file is deleted when you terminate the session. If the editing session is aborted for any reason, such as a system failure, CTRL/Y, or the QUIT/SAVE command, you can recover your edits, except for the last few keystrokes. To recover an editing session, type the command you used to invoke the EDT Keypad Emulator with the /RECOVER qualifier. For example, suppose you began an editing session with the command:

  ```
  $ EDIT/TPU/SECTION=EDTSECINI LETTER.RNO
  ```

  To recover that editing session, enter the following command and file name:

  ```
  $ EDIT/TPU/RECOVER/SECTION=EDTSECINI LETTER.RNO
  ```

  Note

  If you accidentally type the TJL file type, the EDT Keypad Emulator will display the contents of the journal file on your screen. Terminate the editing session with the QUIT command.

There are two restrictions to the journaling facility; one involves the WRITE command, and the other, CTRL/C.

- WRITE command restriction—If you use the WRITE command during your editing session to copy the contents of the buffer to a file, you must recover the original version of the file that you were editing. For example, if you are creating a file called LETTER.RNO;1 and you use the WRITE command without supplying a file specification, the EDT Keypad Emulator creates a file named LETTER.RNO;2 in your current default directory. If you experience a system interruption, you must specify the original version of LETTER.RNO when you recover the file. In this example, the original version is LETTER.RNO;1. See Section 2.4.1 for more information on the WRITE command.

- CTRL/C restriction—The EDT Keypad Emulator may not be able to recover keystrokes entered after you press CTRL/C. If you press CTRL/C, immediately EXIT from the editing session, then reinvoke the EDT Keypad Emulator. DIGITAL does not recommend using CTRL/C during EDT Keypad Emulator editing sessions.

2.3 Using Keypad Editing and Control Key Sequences

EDT Keypad Emulator keypad commands are very similar to EDT keypad commands. The following table lists the two keypad commands that work differently in the EDT Keypad Emulator.

Command Name	Keys to Press	EDT Keypad Emulator Function	EDT editor Function
COMMAND	PF1 and 7	Invokes the prompt **TPU Command:** at which you can type VAXTPU commands and one-line VAXTPU programs, but not EDT Keypad Emulator line editing commands. Press CTRL/Z to invoke the asterisk prompt (*) at which you can type EDT Keypad Emulator line editing commands.	Invokes the prompt **Command:** at which you can type EDT line editing commands.
HELP	PF1 and PF2	Invokes the prompt **Topic:** for help, at which you can type a question mark (?) for a list of Help Utility topics.	Invokes the first level of help text for the EDT editor.

The EDT Keypad Emulator uses the same control key sequences as the EDT editor with one additional control key sequence (CTRL/F). The following table lists the control key sequences that work differently in the EDT Keypad Emulator.

Control Key	EDT Keypad Emulator Function	EDT Editor Function
CTRL/C	Stops execution of an EDT Keypad Emulator command and invalidates the journal file.	Stops execution of an EDT command.
CTRL/F	Resumes an editing session after you press CTRL/Z to exit from help.	None
CTRL/K	Executes the DEFINE KEY command, which accepts only VAXTPU syntax; first prompts for the key definition and then for the key to define.	Executes the DEFINE KEY command, which accepts only EDT syntax; first prompts for the key to define and then for the key definition.
CTRL/T	Causes a line of information about your current process to appear in the MESSAGE_BUFFER at the bottom of the screen.	Causes a line of information about the current process to appear in a random location on your screen.
CTRL/Z	Invokes a prompt at which you can type EDT Keypad Emulator line editing commands; also exits from the Help Utility.	Invokes a prompt at which you can type EDT line editing commands. Pressing PF1 and 7 on the editing keypad also invokes a prompt at which you can type EDT Keypad Emulator line editing commands.

2.4 EDT Keypad Emulator Line Editing

The EDT Keypad Emulator provides a subset of EDT line editing commands. However, you can use all the VAXTPU commands from within the EDT Keypad Emulator, and you can extend the EDT Keypad Emulator by writing VAXTPU procedures. Before entering an EDT Keypad Emulator line editing command, press CTRL/Z; the asterisk prompt (*) appears at the bottom left corner of your screen. Type the command and parameters, if any, and press RETURN or the ENTER key. (EDT editor line editing commands are always terminated by pressing the ENTER key.) To return to keypad editing, type C (for CHANGE MODE) and press RETURN or ENTER.

Some of the line editing commands are described by function in the following sections. See Appendix G of the *VAX Text Processing Utility Reference Manual* for a complete description of all EDT Keypad Emulator line editing commands.

2.4.1 Reading and Writing Files

Two EDT Keypad Emulator line editing commands can be used to manipulate files during an editing session.

- Reading files—The INCLUDE command copies the specified file from your current default directory into the current EDT Keypad Emulator editing session. (To include a file that's not in your current default directory, give the full file specification.) To include a file in your current editing session, press CTRL/Z, type INCLUDE file-spec, and press RETURN.
- Writing files—The WRITE command creates a new file that contains either the entire contents of a buffer or a selected section of a buffer. To write the entire contents of the current buffer to a file, press CTRL/Z, type WRITE file-spec, and press RETURN. To write the contents of a selected section of the current buffer to a file, highlight a select region on your screen using the SELECT keypad command, press CTRL/Z, type WRITE file-spec SELECT, and press RETURN. To write the contents of a buffer that is not the current buffer to a file, press CTRL/Z, type WRITE file-spec =buffer, and press RETURN.

2.4.2 Substituting Strings

The SUBSTITUTE line editing command substitutes one character string for another. Specify the old string enclosed in backslashes followed by the new string enclosed in backslashes. If you specify the /WHOLE command qualifier when you execute the SUBSTITUTE command, all occurrences of the old string in your buffer are replaced with the new string. If you do not specify the /WHOLE command qualifier, only the first occurrence of the old string on the current line is replaced by the new string.

The following example substitutes every occurrence of the string "this" for the string "the" in the text below.

```
The court will now call the defendent
to the stand.  Place your right hand on
the Bible, and repeat after me.
* SUBSTITUTE/the/this/whole

this court will now call this defendant
to this stand.  Place your right hand on
this Bible, and repeat after me.
```

If the cursor was positioned on the first character in the first line the following example replaces the first occurrence of the string "this" in the current line with the string "This".

```
this court will now call this defendant
to this stand.  Place your right hand on
this Bible, and repeat after me.
*SUBSTITUTE/this/This

This court will now call this defendant
to this stand.  Place your right hand on
this Bible, and repeat after me.
```

2.4.3 Controlling Editing Functions with the SET commands

The SET commands control some aspects of the EDT Keypad Emulator's behavior during an editing session. The following list gives the SET commands and their functions. Each SET command has a corresponding SHOW command; see Appendix G of the *VAX Text Processing Utility Reference Manual* for a list of SHOW commands.

- SET CURSOR top:bottom—Determines the lines at which scrolling begins, where top is the upper limit and bottom is the lower. The default values for *top:bottom* are 7:14. Values for the top and bottom can range from 1 to 21. To set the screen to scroll upward at line 2 and downward at line 15, for example, enter the following command line at the asterisk prompt:

  ```
  *SET CURSOR 2:15
  ```

- SET SCREEN width—Sets the maximum number of characters that the EDT Keypad Emulator displays on a line of text on your terminal screen; by default, *width* is 80. The maximum value for *width* on all VT200s or on VT100s with advanced video option is 132. The maximum value for *width* on a VT100 without the advanced video option is 80. To set the screen width on your terminal to 132, for example, enter the following command line at the asterisk prompt:

  ```
  *SET SCREEN 132
  ```

- SET SEARCH setting—The values that you provide for *setting* determine how the EDT Keypad Emulator performs a search with the FIND keypad command. The possible values for *setting* are:

  GENERAL or EXACT—If you specify GENERAL, the default, the EDT Keypad Emulator ignores the case of the letters and diacritical marks in the search string when it performs the search. If you specify EXACT, the case of each letter in the search string (uppercase or lowercase) and diacritical marks (accents) are matched exactly during a search.

  BEGIN or END—If you specify BEGIN, the default, the cursor stops on the first character in the found string. If you specify END, the cursor stops at the character immediately following the last character in the found string.

- SET TAB number—Allows you to change the first tab value. The EDT Keypad Emulator moves the cursor to this tab value only if the cursor appears at the left margin of the screen when the TAB key is pressed. Values for *number* can range from 0 to 255. By default, tabs are set every eight spaces. For example, to set the first tab value equal to 10, enter the following:

```
* SET TAB 10
```

- SET WRAP number—Specifies the cursor position at which the EDT Keypad Emulator will automatically wrap text to the next line. Values for *number* can range from 0 to 255 (0 turns wrap off). When you are inserting text and the cursor position exceeds the value of *number*, the EDT Keypad Emulator wraps the next full word to the next line. However, when you insert text in the middle of a line, that line will not always wrap. To wrap a word that exceeds the seventieth cursor position, for example, enter the following:

```
* SET WRAP 70
```

2.5 Differences Between the EDT Keypad Emulator and the EDT Editor

The following list outlines some of the differences between the EDT Keypad Emulator and the EDT Editor.

- Batch Jobs—The EDT editor may not be run in a batch job. The EDT Keypad Emulator may be run in a batch job and can read batch log files while the job is still in progress if you enter the /NODISPLAY qualifier on the EDIT/TPU command line. See Appendix G of the *VAX Text Processing Utility Reference Manual* for a complete description of the EDIT/TPU command and its qualifiers.
- Callable VAXTPU—There are differences between callable VAXTPU and callable EDT. For a complete discussion of callable VAXTPU, see the *VAX Text Processing Utility Reference Manual.*
- Command Termination—Terminate an EDT editor line editing command by pressing the ENTER key. Press the ENTER or the RETURN key to terminate EDT Keypad Emulator line editing command.
- Control Characters—The EDT editor and EDT Keypad Emulator display control characters for an escape sequence, carriage return, line feed, form feed, and vertical tab differently on the screen. For all control characters except the escape sequence, the EDT Keypad Emulator inserts a special visible graphics character into the file.
- CTRL/C—Pressing CTRL/C during an EDT editor session simply stops the execution of an EDT command. Pressing CTRL/C during an EDT Keypad Emulator session not only stops the execution of a command, but may also invalidate the journal file. DIGITAL recommends that you immediately exit from an EDT Keypad Emulator editing session if you press CTRL/C.
- Defining Keys—The syntax of the EDT Keypad Emulator DEFINE KEY command is the reverse of the syntax of the EDT editor DEFINE KEY command. When defining keys with the EDT Keypad Emulator DEFINE KEY command, you must first specify the key definition and then specify the key to define.
- Error Messages—The EDT editor issues EDT error messages, and the EDT Keypad Emulator issues VAXTPU error messages. All VAXTPU error messages issued during an editing session remain in the MESSAGE BUFFER; you can move the cursor to the message buffer to review error messages at any time in the editing session.
- Journal Files—Journal files created by the EDT editor have a file type of JOU. Journal files created by the EDT Keypad Emulator have a file type of TJL.

- Maximum File Size—When you are using the EDT editor, only part of the file you are editing is memory resident in a work file. EDT Keypad Emulator files, however, are memory resident; they must be small enough to fit into your virtual address space. Startup of an EDT Keypad Emulator editing session requires the time that it takes to read the entire file into main memory.
- Messages—EDT displays messages randomly across the screen; this may cause text to be overwritten or the screen to scroll if messages are written near the bottom of the screen. The EDT Keypad Emulator displays messages at the bottom of the screen, and text that appears on the screen is not affected.
- VT52 Terminals—EDT is fully supported on VT52 terminals. VAXTPU does not support screen oriented features on a VT52.
- "Working" Message—EDT displays a "working" message in reverse video at the bottom middle of the screen. VAXTPU has a user-definable TIMER message that is not automatically defined by either of the editing interfaces. The following TPU command defines a "working" message for an EDT Keypad Emulator editing session:

```
TPU Command: SET (TIMER, ON, "Working...");
```

The following section describes how to enter VAXTPU commands during an EDT Keypad Emulator editing session.

2.6 Using VAXTPU Commands from Within the EDT Keypad Emulator Interface

You can enter a VAXTPU command or a series of VAXTPU commands that does not exceed one line at the prompt TPU Command:. To invoke the TPU Command: prompt, press the PF1 and 7 keys on the numeric keypad. For example, to execute the APPEND_LINE command, which places the current line at the end of the previous line, press the PF1 and 7 keys, type APPEND_LINE, and press RETURN. The *VAX Text Processing Utility Reference Manual* contains a complete list of VAXTPU commands.

Table 2.1 provides a brief list of EDT line editing commands and their corresponding VAXTPU commands.

Table 2–1 VAXTPU EDT Keypad Emulator Line Commands

EDT	VAXTPU
CLEAR main_buffer	DELETE(main_buffer)—to remove buffer + text
COPY =main TO =paste	POSITION(paste_buffer) COPY_TEXT(main_buffer) POSITION(main_buffer)
DEFINE KEY key_id as command	DEFINE_KEY (command KEY_NAME(key_id))
DELETE =main	ERASE(main_buffer)—to remove text only
EXIT	EXIT
FILL =main	FILL(main_buffer,' ',r_margin,l_margin)
FIND =my_buffer or =my_buffer	buffer_name := CREATE_BUFFER('my_buffer') MAP(main_window,buffer_name)
INCLUDE foo.bar	READ_FILE('foo.bar')
INSERT;character_string	COPY_TEXT('character_string')
QUIT	QUIT
SET CURSOR 7:14	SET (SCROLLING, CURRENT_WINDOW, 6, 7, 0)
SET SCREEN 132	SET(WIDTH,main_window,132)
SHOW BUFFER	SHOW(BUFFERS)
SHOW SCREEN	SHOW(SCREEN) SHOW(WINDOWS)
SHOW VERSION	SHOW(SUMMARY)
SHL	SHIFT(CURRENT_WINDOW, 8)
SHR	SHIFT(CURRENT_WINDOW, -8)
WRITE foo.bar	WRITE_FILE(main_buffer,'foo.bar')

2.7 Extending the VAXTPU EDT Keypad Emulator Interface

The VAXTPU EDT Keypad Emulator is written in the VAXTPU programming language. You can extend the EDT Keypad Emulator's functions by writing procedures in the VAXTPU programming language, compiling them, and saving them in command or section files. Key definitions and executable statements can also be saved in command or section files.

DIGITAL strongly recommends that you study the EDT Keypad Emulator source code, which is stored in SYS$SHARE:EDTSECINI.TPU, before you begin writing procedures to modify the EDT Keypad Emulator interface. You should ensure that variables and statements in your procedures are consistent

with the EDT Keypad Emulator's variables and statements, to avoid making changes that will negatively affect the EDT Keypad Emulator's operations.

You must be familiar with programming and programming concepts before attempting to customize the VAXTPU EDT Keypad Emulator editing interface. The following sections assume you are familiar with the VAXTPU language that is described in the *VAX Text Processing Utility Reference Manual*.

2.7.1 Writing VAXTPU Procedures

If you are combining several VAXTPU language statements to perform a text processing operation, create the VAXTPU program by entering the text into a buffer. To enter a sample procedure, invoke the VAXTPU EDT Keypad Emulator as follows:

```
$ EDIT/TPU/SECTION=EDTSECINI
```

You are now ready to create a new file. The following sections tell you how to write, compile, and save a procedure that allows you to create a second window on your terminal screen.

Enter the following text into the main buffer.

```
PROCEDURE TWO_FILES
extra_file   := READ_LINE ("File name: "); ! ask for file name
extra_buf    := CREATE_BUFFER ("EXTRA", extra_file);
extra_window := CREATE_WINDOW (1, 11, ON);
ADJUST_WINDOW (main_window, +11, 0);    ! Reduce the main window
SET (STATUS_LINE, extra_window, REVERSE, "EXTRA WINDOW");
MAP (extra_window, extra_buf);
ENDPROCEDURE
```

This procedure performs the following actions when it is invoked:

1. Prompts for an input file name
2. Creates a buffer and reads in the specified file
3. Creates a new window, using the top 11 lines of the screen, and adjusts the existing main window to accommodate it
4. Defines a status line for the bottom of the new window
5. Associates the new buffer with the new window and maps the new window to the screen

2.7.2 Compiling a Procedure

Before you can use a VAXTPU procedure, you must compile it. To do this, press the PF1 and 7 keys on the numeric keypad and enter the COMPILE command. For example, to compile the procedure named TWO_FILES, press the PF1 and 7 keys and enter the following TPU command (be sure to include the parentheses):

```
TPU Command: COMPILE (MAIN_BUFFER)
```

After compiling a procedure, execute it by simply invoking it. Press the PF1 and 7 keys on the numeric keypad and enter the procedure name on the command line. For example, to execute the procedure named TWO_FILES, press the PF1 and 7 keys on the numeric keypad and enter the following at the TPU command prompt:

```
TPU Command: TWO_FILES
```

To make sure that the procedures and key definitions you create in the current session are available in future sessions, you must save them. The following sections describe how to save procedures in command and section files.

2.7.3 Saving Procedures in Command and Section Files

You can save procedures and key definitions in command and section files. A command file is a VAXTPU source code file that may contain VAXTPU procedures, key definitions, and other VAXTPU statements. When you invoke the EDT Keypad Emulator with the /COMMAND=file-spec qualifier, the command file is compiled and executed; all procedures and key definitions stored in the command file are available during your editing session. A section file is a collection of compiled VAXTPU procedure definitions, variable definitions, and key definitions. When you invoke the EDT Keypad Emulator with the /SECTION=file-spec qualifier, the section file is executed; all procedures and key definitions stored in the section file are available during your editing session.

2.7.3.1 Using a Procedure as a Command File

To save a procedure in a command file, exit from the editing session in which you created the procedure, saving the contents of the main buffer in a file. For example, to save the procedure named TWO_FILES press CTRL/Z and type the following at the asterisk prompt:

```
* EXIT TWO_FILES_COMMAND.TPU
```

The next time you invoke the EDT Keypad Emulator you can specify a command file; and VAXTPU will compile your procedure during startup. For example, to use TWO_FILES_COMMAND.TPU as a command file, enter the following TPU command line:

```
$ EDIT/TPU/SECTION=EDTSECINI/COMMAND=TWO_FILES_COMMAND.TPU TEXT.RNO
```

To invoke the procedure, press PF1 and 7 and enter the procedure name TWO_FILES on the TPU command line.

```
TPU Command: TWO_FILES
```

2.7.3.2 Adding a Procedure to a Section File

A second method of saving your procedures and key definitions is by adding them to a section file. Like command files, section files contain VAXTPU commands and procedures. Section files, however, are already compiled and are stored in binary form. Using a precompiled section file rather than a command file results in much faster startup when you invoke the EDT Keypad Emulator editor.

Use the VAXTPU SAVE built-in procedure to add procedures and key definitions that you have created during an editing session to a section file. The SAVE built-in procedure creates a section file containing all currently compiled procedures and key definitions. Execute the VAXTPU SAVE built-in procedure after you compile the buffer containing your procedures and key definitions. To use the SAVE built-in procedure, press PF1 and 7 keys on the numeric keypad and enter the following TPU command:

```
TPU Command: SAVE ("SYS$LOGIN:my_editor.TPU$SECTION")
```

where my_editor is a name that you choose for your section file.

You should then use the QUIT or the EXIT command to leave the editor. This ensures that your section file is created properly.

2.8 Defining Keys

Use the VAXTPU DEFINE_KEY built-in procedure to define a key to execute a VAXTPU command automatically. Press PF1 and 7 on the numeric keypad and enter the DEFINE_KEY command at the TPU command prompt, using the following syntax:

```
TPU Command:  DEFINE_KEY ("a vaxtpu program", KEY_NAME (the key to define))
```

You can define all keys on the VT100 and VT200 keyboards and keypads with the following exceptions:

- COMPOSE CHARACTER
- F11 (ESCAPE)
- SHIFT
- F1 through F5
- PF1, the default shift key for the editor. (If you wish to define this key, you must first use the SET (SHIFT_KEY,keyword) built-in procedure to define a different shift key for the editor.)

Although the TWO_FILES procedure creates two windows, you still need to specify a VAXTPU command every time you wish to move the cursor from one window to another. Therefore, you might want to define keys to move you quickly from one window to the next. The following examples show you how to define keys to move the cursor from one editing window to another.

Enter the following text into the main buffer.

```
PROCEDURE TWO_FILES
extra_file   := READ_LINE ("File name: "); ! ask for file name
extra_buf    := CREATE_BUFFER ("EXTRA", extra_file);
extra_window := CREATE_WINDOW (1, 11, ON);
ADJUST_WINDOW (main_window, +11, 0);     ! Reduce the main window
SET (STATUS_LINE, extra_window, REVERSE, "EXTRA WINDOW");
MAP (extra_window, extra_buf);
ENDPROCEDURE
```

Compile this procedure by pressing PF1 and & and entering the following TPU command:

```
TPU Command: COMPILE (current_buffer)
```

Press the PF1 and 7 keys on the numeric keypad and enter the following TPU command:

```
TPU Command: DEFINE_KEY ("POSITION (MAIN_WINDOW)", KEY_NAME (DOWN, SHIFT_KEY))
```

This command binds the PF1/down-arrow key sequence (PF1 is the default shift key) to the command to move the cursor to the current position into the main window. Similarly, you can define the PF1/up-arrow key sequence to move the cursor to the current position to the extra window. Press the PF1 and 7 keys on the numeric keypad and enter the following TPU command:

```
TPU Command: DEFINE_KEY ("POSITION (EXTRA_WINDOW)", KEY_NAME (UP, SHIFT_KEY))
```

To invoke the procedure TWO_FILES, press the PF1 and 7 keys on the numeric keypad and enter the following:

```
TPU Command: TWO_FILES
```

This procedure then prompts you for an input file name. Enter the name of the file. The screen splits, a new buffer is created, and the specified file is associated to the new window. You may now move between the new window and the main window by pressing either the PF1/up-arrow or the PF1/down-arrow key sequences. Save the TWO_FILES procedure and the key definitions for the PF1/down-arrow and the PF1/up-arrow key sequences by pressing the PF1 and 7 keys on the numeric keypad and entering the following TPU command:

```
TPU Command: SAVE ("SYS$LOGIN:MY_EDITOR.TPU$SECTION")
```

Now, whenever you wish to use the EDT interface and split-screen capabilities, you can invoke the EDT Keypad Emulator and specify your personal section file with the DCL command:

```
$ EDIT/TPU/SECTION=SYS$LOGIN:MY_EDITOR.TPU$SECTION
```

Whenever you wish to split the screen and use the associated defined key capabilities, press the PF1 and 7 keys on the numeric keypad and enter the following:

```
TPU Command: TWO_FILES
```

3 Editing Files With VAXTPU (EVE Interface)

EVE, the Extensible VAX Editor, is an editing interface to the VAX Text Processing Utility (VAXTPU). With EVE you can create new files, insert text into them, and edit and manipulate that text. You can also edit and manipulate text in existing files. EVE is interactive; you can see the changes to a file as you make them.

3.1 Invoking and Terminating EVE

An editing session begins when you invoke EVE with the DCL command EDIT/TPU and ends when you terminate EVE with the EXIT or QUIT command. EVE does not destroy the contents of any existing file that you edit; it simply produces a new version of a modified file when you terminate EVE with the EXIT command.

3.1.1 Invoking EVE

You may start an editing session by creating a new file and inserting text into it during the session. You may name this file either when you invoke EVE by typing EDIT/TPU file-name, or when you EXIT from EVE by responding to the question "Enter a file name to write buffer MAIN; else press RETURN:". You may also specify an existing file when you invoke EVE.

When you invoke EVE, the text of the file you are editing is written into a buffer. The contents of the buffer are shown on a section of your screen that is called a window. EVE buffers exist only during the editing session. When you end an editing session, you specify whether the contents of an EVE buffer are to be written to a file or discarded.

To invoke EVE to create a new file named NEWFILE.DAT, enter the following command:

```
$EDIT/TPU NEWFILE.DAT
```

This command produces a screen that appears as follows:

```
[End of file]

Buffer NEWFILE.DAT                    Insert        Forward
Editing new file: could not find WORKDISK:[USER]NEWFILE.DAT
```

The [End of file] marker is always located at the end of an EVE buffer. It is only visible on the screen, and does not become a part of your file when you terminate the EVE editing session. When you add text to the buffer, [End of file] moves downward. The [End of file] marker may not be visible when you are viewing the beginning of a buffer that contains many lines of text.

A highlighted status line appears at the bottom of the EVE window and provides information about the EVE buffer. The buffer name, current mode (Insert or Overstrike), and current direction (Forward or Reverse), are displayed in the status line.

If you invoke EVE with a file name, an informational message appears in the message buffer beneath the highlighted status line stating that either the file is a new file or that a certain number of lines were read from an existing file. EVE communicates with you throughout the editing session by displaying messages in the message window.

To invoke EVE to edit an existing file named OLDFILE.DAT, enter the following command:

```
$ EDIT/TPU OLDFILE.DAT
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert      Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

When you invoke EVE to edit an existing file, you can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the directory wildcard character ([. . .]) as a substitute for a directory specification. If there is only one match, the file is displayed on your screen. If there is more than one match, EVE displays a list of choices and prompts you to provide a more complete file specification. If there is no match, BUFFER_MAIN appears in the highlighted status line and the message "No files matching:" is displayed in the message buffer below.

You can specify some optional parameters to the EDIT/TPU command when you invoke EVE. See the appropriate appendix of the *MicroVMS User's Manual* for a complete description of the EDIT/TPU command.

You may wish to use a command symbol to invoke the EVE editing interface. For example, if you place the following statement in your login command file and execute your login command file, you will only need to type EVE to invoke the EVE editor.

```
$ EVE == "EDIT/TPU"
```

3.1.2 Terminating EVE

Both the EXIT and QUIT commands terminate an EVE editing session. EXIT saves your edits in a new version of the file; QUIT does not automatically save edits. Any existing versions of the files remain unchanged regardless of how the editing session is terminated.

3.1.2.1 Saving Your Edits

To save your edited text, use the EXIT command to terminate EVE. Enter the EXIT command by pressing the F10 key (on a VT200-series terminal) or by pressing CTRL/Z. If you have modified the contents of a file, EVE creates a new version of the file with the same file name and file type as the original version and the version number incremented by 1. For example, if you use the EXIT command after modifying a file named FUN.DAT;1, the output file is named FUN.DAT;2.

```
Buffer FUN.DAT                                    Insert      Forward
4 lines written to file WORKDISK:[USER]FUN.DAT;2
$
```

3.1.2.2 Discarding Your Edits

To terminate EVE without saving your edits, press the DO key, type QUIT, and press RETURN. If you have modified a file named FUN.DAT, the following display appears on your terminal screen:

```
Buffer FUN.DAT                                    Insert      Forward
Buffer modifications will not be saved, continue quitting (Y or N)?
```

Type Y and press RETURN if you want to quit without saving the edits. If you change your mind and decide to save your edits, type N, press RETURN, and exit from the file using the EXIT command.

3.2 Using the Help Facility

EVE's online help facility allows you to get help during your editing session without disturbing your work. Press the DO key and enter HELP to view a portion of a list of EVE commands. Press the Prev Screen and Next Screen keys to scroll through the entire list of EVE commands. When you type a command name and press RETURN, the help text appears on the screen.

If you know the name of a specific command for which you need help, press the DO key, enter HELP followed by the name of the command, and press RETURN. The help text for that command appears on the screen. For example, to receive help on the TOP command, press the DO key, enter HELP TOP, and press RETURN. The following help text appears on your screen:

```
TOP
   TOP moves the cursor to the beginning of the current buffer

Help buffer
Type command name, or ? for list (press Return if done):
```

EVE also provides help for the keypad keys. Press the Help key. A list and diagram of editing keys appears on your screen. Press the editing key for which you need help, and the help text for that key appears on your screen.

Note

Although the Help diagram does not display any keys you have defined, EVE provides a message stating that you have defined the key if you press a key to which you have assigned a learn sequence. If you press a key to which you have assigned an EVE command, EVE provides the help text for that EVE command. Section 3.8 explains how to assign learn sequences and EVE commands to keys.

3.3 Recovering from System Interruptions

You can recover from two types of system interruptions to your editing session. Press CTRL/W to remove extraneous characters that appear on your screen and use the /RECOVER qualifier of the EDIT/TPU command to recover a "lost" editing session.

If extraneous characters appear on your terminal screen while you are editing or inserting text, press CTRL/W to refresh your screen. The screen becomes blank, and then all characters are redrawn, minus any extraneous characters.

If you are editing a file and a system interruption (break in communication between your terminal and the computer) occurs, you can recover your "lost" editing session. By default, EVE records every keystroke you enter during an editing session in a journal file that has the same file name as the file you were editing and a file type of TJL. If an editing session ends without interruption, the system deletes the journal file. When you experience a system interruption, however, the journal file is saved and only the last few keystrokes of your editing session are lost.

To recover an editing session, enter the command you used to invoke EVE plus the /RECOVER qualifier. You must recover an editing session at a terminal of the same type as the one you used for your editing session. When EVE finishes recovering the session, check to ensure that the last few keystrokes of your editing session were recovered and continue editing the file. If another system interruption occurs before you exit, a journal file containing the keystrokes from both editing sessions is saved.

Note

If you type the TJL file type by mistake with the EDIT/TPU/RECOVER command, EVE displays the contents of the journal file on your screen. Terminate the editing session with the QUIT command, and enter the EDIT/TPU/RECOVER command with the correct file type.

For example, to recover an editing session you began with the command EDIT/TPU LETTER.RNO, type the following command and file name and press RETURN:

```
$ EDIT/TPU/RECOVER LETTER.RNO
```

There are two restrictions to using the journaling facility; the first restriction involves the WRITE FILE command and the second restriction involves CTRL/C.

First, if you use the WRITE FILE command during your editing session to copy the contents of the buffer to another file, you must recover the original version of the file that you were editing. If you do not specify the original version number, you will not be able to recover your file. For example, if you are editing an existing file called LETTER.RNO;1 and use the WRITE FILE command, EVE creates LETTER.RNO;2. If you experience a system interruption, you must enter the original version of LETTER.RNO on the EDIT/TPU/RECOVER command line. In this example it would be LETTER.RNO;1. See Section 3.6.3 for more information on the WRITE FILE command.

Second, EVE may not be able to recover keystrokes entered after you press CTRL/C. If you press CTRL/C, immediately EXIT from the editing session, then reinvoke EVE. DIGITAL does not recommend using CTRL/C during EVE editing sessions.

3.4 Entering EVE Commands

Enter EVE commands by pressing editing keys, or by pressing the DO key (VT200) or PF4 (VT100) and typing a command at the "Command:" prompt.

3.4.1 Pressing Editing Keys to Enter EVE Commands

EVE defines some editing keys automatically and you may define additional editing keys (see Section 3.8). The automatically defined editing keys on VT200-series terminals include the minikeypad and some function keys and control key sequences. On VT100-series terminals, EVE automatically defines some of the numeric keypad keys, the arrow keys, and some control keys. Press an editing key to enter an EVE command. Each automatically defined editing key performs only one editing command.

Throughout this chapter, EVE editing keys are referred to by their names, rather than by their location on the VT200-series or VT100-series keyboards. For example, on a VT200-series terminal, the DO key is located at the top of the editing keypad and is labeled DO. On a VT100-series terminal, the DO key is located at the upper left of the numeric keypad and is labeled PF4. Figure 3–1 shows the keys that are automatically defined on the VT200-series terminal and Figure 3–2 shows the keys that are automatically defined on the VT100 series terminal.

Figure 3–1 Editing Keys—VT200-Series Terminals

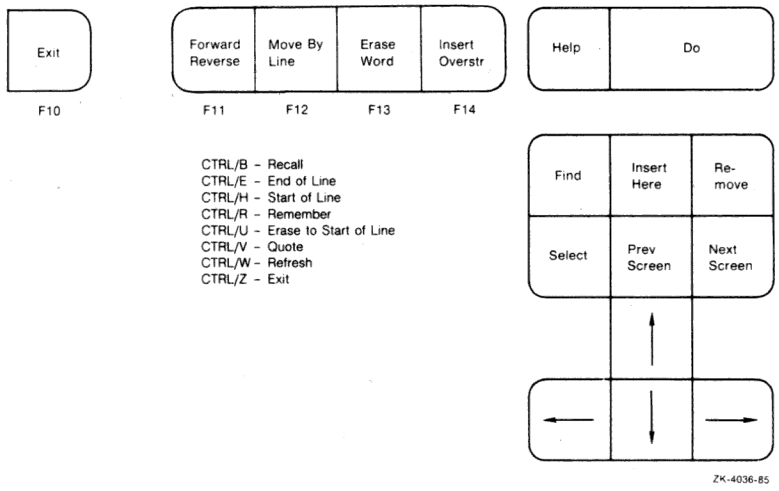
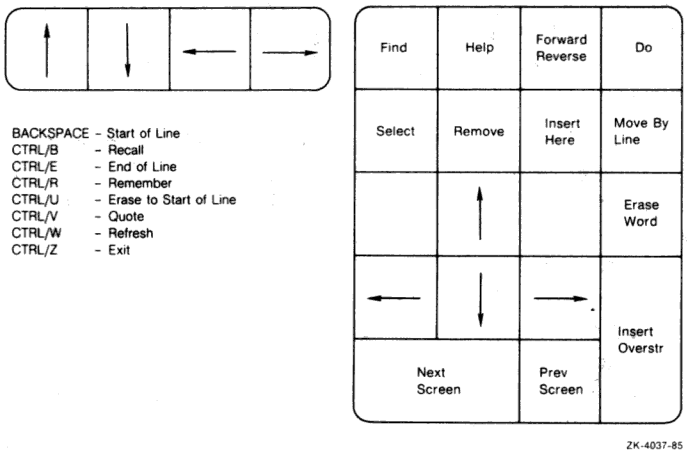


Figure 3–2 Editing Keys—VT100-Series Terminals



Note that EVE uses the numeric keypad differently on VT100 and VT200 terminals. EVE automatically defines 16 of the 18 numeric keypad keys on the VT100 as editing keys, but it does not automatically define the VT200 numeric keypad keys as editing keys. You can use the VT200 numeric keypad keys to enter numeric data, or you can define them to enter EVE commands or learn sequences (see Section 3.8).

3.4.2 Entering EVE Commands at the Command: Prompt

Press the DO key to invoke the "Command:" prompt. Type the EVE command that you wish to execute and press RETURN. You can fix typing mistakes on the EVE command line by pressing DCL line editing keys. By default, the editing mode of the EVE command line is overstrike, but if you have issued the SET TERMINAL/INSERT command at the DCL command level before invoking EVE, the EVE command line will be in insert mode.

To save keystrokes when you are typing EVE commands, you can use CTRL/B, abbreviate commands, and use the REPEAT command. Each method of saving keystrokes is described as follows:

- Press CTRL/B once to recall the last EVE command that you typed. Pressing CTRL/B again recalls the previous command that you typed. Continue pressing CTRL/B until the command you wish to execute appears on your screen, and press RETURN to execute the command.

- Abbreviate EVE command names, as long as the abbreviation is unique. If you type an abbreviation that is not unique, a list of choices appears in the message buffer immediately below the highlighted status line. Type enough additional characters to ensure that the abbreviation is unique, and press RETURN to execute the command. You can also abbreviate buffer names, file names, mark names, and procedure names. Again, EVE provides a list of choices if you do not provide a unique abbreviation.
- Use the REPEAT command to repeat an EVE command or learn sequence or to insert a printing character a specific number of times. For example, to insert the character *p* in your editing buffer 20 times, press the DO key, type REPEAT 20, and press RETURN. Type *p*. EVE inserts it into the current buffer 20 times.

  To repeat an EVE command or learn sequence, press the DO key, type REPEAT n, and press RETURN. To execute an EVE command or learn sequence that is bound to an editing key n times, simply press the key. If the EVE command is not bound to a key, press the DO key, type the EVE command to be repeated, press RETURN, and EVE executes the command n times. Pressing the DO key twice activate the last command issued.

3.5 Editing Files with EVE

Use EVE commands to move the cursor, insert text, erase and restore text, move text from one location to another, and locate text strings.

3.5.1 Moving the Cursor

The following table shows the editing keys and EVE commands that move the cursor.

Key Name	Destination of Cursor
UP arrow	Moves the cursor up one character.
LEFT arrow	Moves the cursor one character to the left.
DOWN arrow	Moves the cursor down one character.
RIGHT arrow	Moves the cursor one character to the right.
CTRL/E	Moves the cursor to the end of the current line.
CTRL/H	Moves the cursor to the beginning of the current line.

Key Name	Destination of Cursor
Move By Line	If the current direction is forward, moves the cursor to the end of the current line or to the end of the next line if the cursor is already at the end of a line. If the current direction is reverse, moves the cursor to the beginning of the current line or to the beginning of the previous line if the cursor is already at the beginning of a line.
Prev Screen	Moves the cursor to the previous screen of the current buffer.
Next Screen	Moves the cursor to the next screen of the current buffer.

Command Name	Destination of Cursor
BOTTOM	Moves the cursor to the end of the current buffer.
BUFFER	Puts the specified buffer in the current window, and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
GET FILE	Creates a new buffer that contains the text of the specified file (or an empty buffer if you specify a file that does not exist), places the new buffer to the current window, and places the cursor at the beginning of the new buffer. If you specify the same file again during an editing session, GET FILE simply places the buffer in the current window. If you specify the same file name and file type with a different device or directory name during an editing session, EVE prompts you for a buffer name into which to read the file.
LINE	Moves the cursor to the beginning of line n in the current buffer, where n is a positive integer.
MOVE BY WORD	If the current direction is forward, moves the cursor to the beginning of the next word. If the current direction is reverse and the cursor is not at the beginning of a word, the cursor moves to the beginning of the current word; otherwise the cursor moves to the beginning of the previous word.
OTHER WINDOW	If there are two editing windows on your screen, moves the cursor to the last location it occupied in the other editing window.
TOP	Moves the cursor to the beginning of the current buffer.

The following example demonstrates how to move the cursor through a buffer.

Invoke EVE to edit the existing file OLDFILE.DAT. (This example assumes that you created this file in an earlier editing session.) EVE reads the contents of OLDFILE.DAT into a buffer and places the cursor at the beginning of the first line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Press CTRL/E to move the cursor to the end of the first line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Press the DO key, type BOTTOM, and press RETURN to move the cursor to the end of the buffer.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Forward
```

Press the Up arrow to move the cursor to the beginning of the fourth line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Forward
```

Press the Forward Reverse key to change the current buffer direction to reverse. Press the Move By Line key, to move the cursor to the beginning of the third line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Reverse
```

Press the DO key, type LINE 1, and press RETURN to move the cursor to the beginning of the first line of text.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

3.5.2 Inserting Text

You can insert text, entire files, and special nonprinting characters (such as control characters) into the current buffer. The following table shows the editing keys and EVE commands that you use while inserting text.

Key Name	Result
Insert Overstr	Changes the current editing mode, which is displayed in the highlighted status line. In insert mode, text is inserted at the current character position, and text that currently appears in the buffer moves to accommodate your insertions. In overstrike mode, text is inserted at the current character position, and text that currently appears in the buffer is overwritten.
CTRL/V	Executes the QUOTE command, which allows you to insert special nonprinting characters such as control characters, and printing characters such as punctuation marks in your buffer. You can use CTRL/V when using the FIND command to search for special characters.

Command Name	Result
INCLUDE FILE	Inserts the entire contents of the specified file into a buffer at the line before the current cursor location.

Before you begin inserting text into a buffer, look at the highlighted status line to determine whether EVE is in insert or overstrike mode. If EVE is in insert mode, text is inserted at the cursor position, and text that already appears in the file moves to accommodate your insertions. If EVE is in overstrike mode, text that you type at the keyboard is inserted at the cursor position, and the text that already appears in the file is overwritten as the cursor moves through it.

Press the Insert Overstr key to change from insert to overstrike modes, and vice versa.

You can insert:

- Text—Type characters at the keyboard to insert text into the buffer at the current cursor position.
- Files—Press the DO key, type INCLUDE FILE, and press RETURN. Type the file specification at the "File to include:" prompt and press RETURN. You can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the directory wildcard character ([. . .]) as a substitute for a directory specification. EVE disregards the current mode (insert/overstrike) of the buffer, and inserts the entire contents of the specified file into the buffer just before the line in which the cursor currently appears. If there is more than one match for a wildcarded file specification, EVE displays a list of choices and prompts you to provide a more complete file specification. If the specified file does not exist, EVE displays a message stating that it could not include the file.
- Special Nonprinting Characters—Execute the QUOTE command by first pressing CTRL/V, and then pressing the special nonprinting character. For example, to insert a form feed into the buffer, press CTRL/V and then press CTRL/L.

The following example shows you how to insert text into a file, first in insert mode and then in overstrike mode. Invoke EVE to edit the existing file OLDFILE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer  OLDFILE.DAT                                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Check the highlighted status line to ensure that EVE is in insert mode. Press the Insert Overstr key to change to insert mode if EVE is in overstrike mode. Move the cursor to the letter *s* in the word supervisor, type Engineering, and press the space bar.

The word Engineering is inserted in your text buffer, and the rest of the text in the line shifts to the right.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer  OLDFILE.DAT                                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Now press the Insert Overstr key to change to overstrike mode. Move the cursor to the letter *D* in the word Donna and type Andrea.

The word Andrea is inserted in the buffer, overwriting the word Donna.

```
Schedule for 1 July
10:00 AM meeting with Engineering supervisor
Read and review memo from Andrea
Work on Pascal program
[End of file]

Buffer  OLDFILE.DAT                                    Insert        Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

3.5.3 Erasing and Restoring Text

The following table shows the editing keys and EVE commands that erase and restore text.

Key Name	Effect
<X] (DELETE)	\Erases the character to the left of the cursor.
Erase Word	Erases the current word or, if the cursor is not on a word, erases the next word.
CTRL/U	Erases all characters from the current cursor position to the beginning of the line.
Select	Marks text for removal (highlights it in reverse video) from the initial cursor location to wherever you move the cursor. If you decide not to remove the text, cancel the selection by pressing the Select key again.
Remove	Removes the text that was marked with Select, placing it in the Insert Here buffer.

Command Name	Effect
ERASE CHARACTER	Erases the current character.
ERASE LINE	Erases from the current cursor position to the end of the current line, appending the next line to the end of the current line.
ERASE PREVIOUS WORD	Erases the current word, unless the cursor is on the initial character or between words, in which case the previous word is erased. If the cursor is at the start of a line, no word is erased and the current line is appended to the previous line.
RESTORE	Restores, at the current cursor position, the text that you have erased most recently with an EVE command or editing key.

To erase text from your buffer, move the cursor to the location of the text that you wish to erase, and press the appropriate editing key or type the appropriate EVE command. The following example shows you how to erase and restore text. Invoke EVE to edit the existing file RHYMES.DAT. (This example assumes that you created this file in an earlier editing session.) Move the cursor to the letter *b* in the word bee.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
```

Press the DO key, type ERASE LINE, and press RETURN.

EVE erases all characters from the letter *b* to the end of the line, and appends the next line to the current line.

```
She rhymes with tree,
also with and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
```

Press the DO key, type RESTORE, and press RETURN, to restore the text you have most recently deleted.

Move the cursor to the letter *y* in the word rhymes.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
```

Press the Erase Word key. EVE erases the word rhymes from your text buffer.

```
She with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
```

Press the DO key, type RESTORE, and press RETURN to restore the word rhymes to the buffer.

Section 3.5.4 describes the functions of the Select and Remove keys, which can be used together to erase text from a buffer.

3.5.4 Moving Text from One Location to Another

The following table describes the functions of the Select, Remove, and Insert Here keys, which are used to erase text and to move text from one location to another within a buffer. Section 3.6 contains guidelines for moving text from one buffer to another.

Editing Key	What it Does
Select	Marks text for removal (highlights it in reverse video) from the initial cursor location to wherever you move the cursor. If you decide not to remove the text, cancel the selection by pressing the Select key again.
Remove	Removes the text that was marked with Select, and places it in the Insert Here buffer.
Insert Here	Inserts the text from the Insert Here buffer at the current cursor location.

To erase text, place the cursor on the first character that you wish to erase. Press the Select key, and move the cursor to the last character that you wish to erase. The text that will be erased is highlighted in reverse video. (If you decide not to remove text from the buffer, press the Select key again to cancel the selection). Press the Remove key. EVE removes the highlighted text from your screen and places it in the Insert Here buffer.

You can either erase text permanently from your buffer by leaving it in the Insert Here buffer, or you can insert the text at any cursor location by pressing the Insert Here key. You may insert the text contained in the Insert Here buffer any number of times at any cursor location until you select a new section of text and place it in the Insert Here buffer using the Remove key.

The following example shows you how to erase and move text from one location to another using the Select, Remove, and Insert Here keys. Invoke EVE to edit the file RHYMES.DAT. Move the cursor to the beginning of the second line of RHYMES.DAT and press the Select key.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
Selection started. Press Remove when finished.
```

Press the down arrow key twice. The second and third lines of text are highlighted. Press the Remove key. The second and third lines of text are removed from the current buffer.

```
She rhymes with tree,
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Remove completed.
```

Do not move the cursor and press the Insert Here key. The text is reinserted in its previous location.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Remove completed.
```

Move the cursor to the end of the third line of text, press RETURN twice, and press the Insert Here key. The text in the Insert Here buffer is inserted at the current cursor location.

```
She rhymes with tree,
also with bee,
and this one makes three.

also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Remove completed.
```

3.5.5 Locating Text

The Find key is used to locate specified strings of text in your buffer. Press the Find key and type the string that you wish to locate, called the search string. Press RETURN, and EVE attempts to move the cursor to the beginning of the specified string. If the search string contains all lowercase letters, EVE disregards the case of letters and diacritical marks (accents) and locates any occurrence of the string. Thus, the, THe, thE, Thë, and THé all match the search string *the*. If the search string contains one or more uppercase letters, EVE locates only the occurrences of the string in which the case of letters and diacritical marks are exactly the same. Therefore, the only match for the search string *tHis* is tHis.

The current direction of the buffer determines whether EVE first attempts to move the cursor in the forward or reverse direction. If EVE finds the string in the current direction and you want to move the cursor to the next occurrence of the string in the current direction, press the Find key twice.

If EVE cannot find the string in the current direction but finds it in the opposite direction, it prompts you about whether you want the cursor to move in the opposite direction. If you type YES or press RETURN, the current direction in the highlighted status line is changed and EVE moves the cursor to the first occurrence of the string in the opposite direction.

The following example uses the existing file RHYMES.DAT to illustrate the use of the Find key. When you invoke EVE to edit RHYMES.DAT, the cursor appears on the first letter of the first line of the buffer, and the current direction is forward.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the Find key, type the letters ree, and press RETURN. The cursor moves to the letter *r* in the word tree.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Forward Find: ree
```

Press Find twice to find the next occurrence of the string *ree*. The cursor moves to the letter *r* in the word three.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Finding previous target: ree
```

3.5.6 Marking Locations in Text

The MARK and GO TO commands are very useful when you are editing a large file and you know that you will want to return to a specific cursor location later in the editing session. The following table describes the MARK and GO TO commands.

Command Name	What it Does
MARK	Associates a unique and invisible label (one or more alphanumeric characters) with the current cursor location for the rest of an editing session.
GO TO	Returns the cursor to the location labeled using the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer, and places the buffer in the current window.

To mark a cursor location, press the DO key, type MARK label-name, and press RETURN. The label name can be one or more alphanumeric characters. To return the cursor to the marked location, press the DO key, type GO TO label-name, and press RETURN.

The following example shows you how to use the MARK and GO TO commands to label a cursor position with the mark FIRST, and to return to that cursor position. Move the cursor to the letter *b* in the word bee. Press the DO key, type MARK and press RETURN. Type FIRST at the highlighted Mark name: prompt to mark the cursor location with the label FIRST.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
Current position marked as FIRST
```

Move the cursor to the letter *t* of the word three.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
Current position marked as FIRST
```

Press the DO key, type GO TO FIRST, and press RETURN to return the cursor to the position labeled FIRST.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
Going to mark FIRST
```

3.5.7 Replacing Text

The REPLACE command allows you to replace a text string that appears in the current buffer with another text string. This is especially useful if you have spelled a word incorrectly throughout a long file, and you wish to fix every occurrence of the incorrectly spelled word.

To use the REPLACE command, press the DO key, type REPLACE, and press RETURN. Type the string that you wish to replace at the highlighted Old string: prompt and press RETURN. Type the new string at the highlighted New string: prompt and press RETURN. If EVE finds the old string in the current direction, EVE moves the cursor to the first occurrence of the old string, highlights the string, and provides the following prompt: "Replace? Type yes, no, all, last, or quit:". If EVE does not find the string in the current direction but does find the string in the opposite direction, it provides the following prompt: "Found in 'reverse/forward' direction. Go there?". If you type yes, the current direction in the highlighted status line is changed. EVE moves the cursor to the first occurrence of the string in the new current direction, highlights the string, and provides the following prompt: "Replace? Type yes, no, all, last, or quit:"

Respond to the prompt by typing a single-character abbreviation of the response, and pressing RETURN. Simply pressing RETURN is equivalent to pressing *y*. The following table lists possible responses and EVE's actions when you type these responses.

Response	EVE's Action
Yes	Replaces the string and attempts to locate another occurrence of the string in the current direction. If found, the cursor moves to the next occurrence of the string, the string is highlighted, and EVE prompts: "Replace? Type yes, no, all, last, or quit:". If the string is not found in the current direction but is found in the opposite direction, EVE prompts: "Found in 'reverse/forward' direction. Go there?".
No	Does not replace the string, and attempts to locate another occurrence of the string in the current direction. If found, the cursor moves to the next occurrence of the string, the string is highlighted, and EVE prompts: "Replace? Type yes, no, all, last, or quit:". If the string is not found in the current direction but is found in the opposite direction, EVE prompts: "Found in 'reverse/forward' direction. Go there?".
All	Replaces the string and all other occurrences of the string in the current direction. EVE moves the cursor to the position where the last replacement occurred. After all occurrences of the string in the current direction have been replaced, EVE may inform you: "Found in 'forward/reverse' direction. Go there?" If you type yes, EVE replaces all occurrences of the string in the new direction.
Last	Replaces this occurrence of the string and stops the REPLACE procedure; the cursor does not move.
Quit	Does not replace this occurrence of the string and stops the REPLACE procedure; the cursor does not move.

The REPLACE command is case sensitive. When both the old string and the new string are lowercase, all case occurrences of the string found in the text are changed and the new string echoes the case of the text it replaces. When the old string is lowercase, but the new string is capitalized or uppercase, the change echoes the case of the new string specified. When the old string is uppercase or capitalized, and the new string is lowercase, uppercase, or capitalized the case of the old string determines which occurrences will be changed, but the change itself echoes the case of the new string.

The following example shows you how to use the REPLACE command to replace every occurrence of the string *ee* with the string *oo*. Look in the highlighted status line to ensure that the current direction is forward. If not, press the Forward Reverse key to change the current direction to forward. Move the cursor to the top of the buffer. Press the DO key, type REPLACE, and press RETURN. Type ee at the highlighted Old string: prompt, press RETURN, and type oo at the highlighted New string: prompt.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Replace? Type yes, no, all, last, or quit:
```

The cursor moves to the highlighted string *ee* in the word tree. Type all, and press RETURN. All occurrences of the string *ee* are replaced with the string *oo*.

```
She rhymes with troo,
also with boo,
and this one makes throo.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Replaced 3 occurrences.
```

3.5.8 Setting Margins, Tabs, and Screen Width

EVE provides commands that enable you to set margins, tabs, and screen width. The following table lists these commands and a description of their functions.

Command Name	Function
SET LEFT MARGIN	Enters new text in the specified column. By default, the left margin is set to 1.
SET RIGHT MARGIN	Enters new text at the end of specified column. EVE wraps words automatically if the right margin setting is less than the width of the screen. If the screen is 80 columns wide, the right margin is 79 by default.
SET TABS AT	Sets tab stops at the columns that you specify as a sequence of positive integers separated by spaces. By default, tab stops are set in every eighth column. SET TABS AT does not affect the hardware tab settings of your terminal.
SET TABS EVERY	Sets tab stops at the specified intervals. By default, tab stops are set in every eighth column. SET TABS EVERY does not affect the hardware tab settings of your terminal.

Command Name	Function
SET WIDTH	Sets the width of lines displayed on the screen. Specify width as a positive integer *n*. If *n* is greater than 80, EVE sets the terminal to 132 character mode. Setting the width changes the text that appears in the current buffer.
SHIFT LEFT	Moves the current window horizontally to the left the specified number of columns. The SHIFT LEFT command can only be used to reverse the effect of the SHIFT RIGHT command.
SHIFT RIGHT	Moves the current window horizontally to the right the specified number of columns, allowing you to view columns of characters that do not currently appear on your terminal screen.

The following example shows you how to use EVE commands to set margins, screen width, and to shift the current window. Invoke EVE to edit the existing file RHYMES.DAT. Press the DO key, type SET LEFT MARGIN 20, and press RETURN to set a left margin of 20. The text that currently appears in the buffer does not change.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Left margin set to 20
```

Move the cursor to the end of the buffer and enter the following new text. The new text that you enter is inserted at the left margin of 20.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
Left margin set to 20
```

Reset the left margin to 1. Press the DO key, type SET LEFT MARGIN 1, and press RETURN. Again, the text that currently appears in the buffer does not change. When you insert new text it is inserted at a left margin of 1.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Left margin set to 1
```

Now set the right margin to 30. Press the DO key, type SET RIGHT MARGIN 30, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Right margin set to 30
```

Enter the following new text in your file and notice that it wraps automatically to the next line at a right margin of 30.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
And free, and fee, and
see, and brie, and any
number of other words.
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Right margin set to 30
```

To reset the right margin to 79, press the DO key, type SET RIGHT MARGIN 79, and press RETURN.

Now set the width of your text window to 20. Press the DO key, type SET WIDTH 20, and press RETURN.

```
She rhymes with tre
also with bee,
and this one makes
                  three.
                  Also with thee, and
                  me.
And free, and fee,
see, and brie, and
number of other wor
[End of file]
Buffer RHYMES.DAT
```

The appearance of the current text window changes, all text beyond the twentieth column disappears from the screen. Press the DO key, type SHIFT RIGHT 5, and press RETURN, to view five columns of text beyond the right boundary of the window.

The window shifts five columns to the right, and you can see characters that were not visible before the shift operation.

```
ymes with tree,
with bee,
his one makes three.
              Also
              me.
ree, and fee, and
and brie, and any
r of other words.
[End of file]

Buffer RHYMES.DAT
Window is now shift
```

Shift the window to its original location by pressing the DO key, typing SHIFT LEFT 5, and pressing RETURN.

```
She rhymes with tre
also with bee,
and this one makes


And free, and fee,
see, and brie, and
number of other wor
[End of file]

Buffer RHYMES.DAT
Window is now shift
```

Set the screen width to 80. Press the DO key, type SET WIDTH 80, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
                  Also with thee, and
                  me.
And free, and fee, and
see, and brie, and any
number of other words.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
```

EVE sets the width of lines on your screen to 80.

3.6 Using Buffers in EVE

Buffers are storage areas that exist only during an editing session. The following table describes the EVE commands that are used to create and manipulate buffers.

Command Name	Effect
BUFFER	Puts the specified buffer in the current window, and moves the cursor to the last location it occupied in that buffer. Creates a new buffer if the specified buffer does not exist.
GET FILE	Creates a new buffer that contains the text of the specified file (or an empty buffer if you specify a file that does not exist), places the new buffer to the current window, and places the cursor at the beginning of the new buffer. If you specify the same file again during an editing session, GET FILE simply places the buffer in the current window. If you specify the same file name and file type with a different device or directory name during an editing session, EVE prompts you for a buffer name into which to read the file.
GO TO	Returns the cursor to the location labeled using the MARK command. If the labeled location is contained in another buffer, EVE moves the cursor to the other buffer, and places the buffer in the current window. Section 3.6.2 explains how to use multiple buffers in an editing session.
SHOW	Displays a screen of information about the current buffer. If more than one buffer is active in your editing session, press the DO key to display information about the other editing buffers.
WRITE FILE	Writes the contents of the current buffer to a file. If you do not specify a file name, EVE uses the buffer name as the file name.

When you invoke EVE to edit an existing file, EVE reads the contents of the file into a buffer. The highlighted status line contains the buffer name, editing mode, and current direction of the buffer. Press the DO key, type SHOW, and press RETURN to display more information about the current buffer. The information includes the buffer name, the name of the input and output files, whether the buffer has been modified, the current mode and direction, the number of lines, the margin and screen width settings, the tab stop settings, and the marks that have been defined in the buffer. If more

than one buffer is active during an editing session, you are prompted to press the DO key to receive information about the other buffers.

3.6.1 Displaying the Contents of the MESSAGES Buffer

EVE uses the MESSAGES window that appears intermittently at the bottom of the screen to communicate error and informational messages during an editing session. EVE also saves these messages in a MESSAGES buffer. Use the BUFFER command to display the contents of the MESSAGES buffer. Press the DO key, type BUFFER MESSAGES, and press RETURN. To return to the buffer that you were editing, press the DO key, type BUFFER buffer-name, and press RETURN.

3.6.2 Using Multiple Buffers in an Editing Session

Use multiple buffers if you want to edit more than one file during an editing session. Multiple buffers are especially useful if you want to copy text from one file to another. To create a new buffer, press the DO key, type GET FILE file-specification, and press RETURN. You can use the asterisk wildcard character (*) as a substitute for all or some of the characters in the file name and file type. You can use the percent wildcard character (%) as a substitute for one character in the file name and file type, and you can use the directory wildcard character ([. . .]) as a substitute for a directory specification.

If the specified file exists, EVE reads the contents of the file into a new buffer and displays the buffer in the current window. If there is more than one match for a wildcarded file specification, EVE displays a list of choices and prompts you to provide a more complete file specification. Otherwise, EVE creates an empty buffer and displays the buffer in the current window.

To change the buffer in the current window, press the DO key, type BUFFER buffer-name, and press RETURN. If you forget a buffer name, use the SHOW command to display the names of active buffers in your editing session.

The following example shows how to use two buffers to edit two files during an EVE editing session. This example assumes that the files RHYMES.DAT and OLDFILE.DAT exist in your current default directory. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
3 lines read from WORKDISK:[USER]RHYMES.DAT
```

Press the DO key, type GET FILE OLDFILE.DAT, and press RETURN to create a new buffer that contains the most recent version of OLDFILE.DAT.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                                    Insert      Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Place the cursor on the letter *R* in the word Read and press the Select key. Press the down arrow once and the third line of OLDFILE.DAT is highlighted. Press the Remove key, to place the highlighted line in the Insert Here buffer.

```
Schedule for 1 July
10:00 AM meeting with supervisor
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                                    Insert      Forward
Remove completed.
```

To resume editing RHYMES.DAT, press the DO key, type BUFFER RHYMES.DAT, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                     Insert      Forward
```

Press the Insert Here key. The text from the Insert Here buffer that was removed from the buffer OLDFILE.DAT is inserted into the buffer RHYMES.DAT.

```
Read and review memo from Donna
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                     Insert      Forward
```

If you exit from an editing session in which you used multiple buffers, EVE writes the contents of the current buffer to a file and asks you whether you want to write the other buffers to files.

3.6.3 Reading Files into Buffers and Writing Files from Buffers

There are three ways to read a file into an EVE buffer. You can:

- Invoke EVE with a file specification.
- Press DO, type INCLUDE FILE file-specification, and press RETURN to read the entire contents of a file into a buffer. EVE inserts the file before the line in which the cursor is currently located.
- Press DO, type GET FILE file-specification, and press RETURN to create a new buffer during an editing session or to read the contents of an existing file into the buffer. (See Section 3.6.2.)

Use the WRITE FILE command to write the contents of the current buffer to a file. Press the DO key, type WRITE FILE, and press RETURN. You can include an optional file specification with the WRITE FILE command. If you do not include a file specification, EVE uses the buffer name as the file name and places the file in your current default directory. If you created the current buffer with the BUFFER command, EVE prompts you for a file specification to which it writes the file.

If you have used the WRITE FILE command in an editing session and you experience a system interruption, see Section 3.3 for information on recovering the editing session.

3.7 Using Multiple Windows

During an EVE editing session, the text buffer you are editing is displayed on the screen in a window. A highlighted status line appears at the bottom of a window identifying the name, current editing mode, and current direction of the buffer. EVE allows you view two windows on your terminal screen at the same time. You can view and edit either two sections of the same buffer (one file) or two different buffers (two files) simultaneously.

The following table describes the EVE commands that are used to create and manipulate windows.

Command Name	What it Does
TWO WINDOWS	Splits the terminal screen and creates two editing windows, moving the cursor to the last position it occupied in the text of the bottom window.
OTHER WINDOW	Moves the cursor to the last position it occupied in the other window.

Command Name	What it Does
ONE WINDOW	Removes the other window from the screen, expanding the current window to occupy the whole editing area of the screen.
GET FILE	Creates a new buffer that contains the text of the specified file (or an empty buffer if you specify a file that does not exist), places the new buffer to the current window, and places the cursor at the beginning of the new buffer. If you specify the same file again during an editing session, GET FILE simply places the buffer in the current window. If you specify the same file name and file type with a different device or directory name during an editing session, EVE prompts you for a buffer name into which to read the file. After you create two windows on your terminal screen, you can use the GET FILE command to create a new buffer in one of your windows.
BUFFER	Puts a new buffer in the current window, and moves the cursor to the last position it occupied in the buffer. Creates a new buffer if the specified buffer does not exist. After you create two windows on your terminal screen, you can use the BUFFER command to put a different buffer in one of the windows.

3.7.1 Editing One File with Two Windows

To view and edit two sections of a file at the same time, press the DO key, type TWO WINDOWS, and press RETURN. EVE splits your screen and creates two windows, moving the cursor to the position it last occupied in the bottom window. Notice that the buffer name in each of the highlighted status lines is the same.

Because you are editing a single file, any edits that you make in one window are made simultaneously in the other window if both windows display the same part of the file. If you are viewing two different sections of a file in the two windows, you may not be able to see the edits being made in the other window. You can select and remove text from one part of the file and insert it into the other. Displaying two sections of a long file makes moving text within a file very efficient. Press the DO key, type OTHER WINDOW, and press RETURN to move the cursor to the other window.

To remove the second window from the screen and expand the current window to occupy the whole editing area, press the DO key, type ONE WINDOW, and press RETURN.

3.7.2 Editing Two Files with Two Windows

The following steps describe how to edit two files with two windows.

1. Invoke EVE to edit a file.
2. Create two windows on your screen by pressing the DO key, typing TWO WINDOWS, and pressing RETURN. EVE splits your screen and creates two windows, moving the cursor to the last position it occupied in the bottom window. Notice that the buffer name in each of the highlighted status lines is the same.
3. Put another file in the current window using the GET FILE or BUFFER command.
   - To create a new buffer in the current window, press the DO key, type GET FILE file-specification, and press RETURN. EVE reads the specified file into the new buffer or creates an empty buffer if the specified file does not exist. If you specify the same file again during an editing session, GET FILE simply places the buffer in the current window. If you specify the same file name and file type with a different device or directory name during an editing session, EVE prompts you for a buffer name into which to read the file.
   - To display a buffer that you created earlier in the editing session in the current window, press the DO key, type BUFFER buffer-name, and press RETURN.
4. You are now viewing two files on your terminal screen. You can select and remove text from one buffer and insert it into the other buffer. Press the DO key, type OTHER WINDOW, and press RETURN to move the cursor to the other window.

The following example shows you how to edit two files and move text from one file to another in two windows. First, invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert        Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

Press the DO key, type TWO WINDOWS, and press RETURN to create two windows on your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward

She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward
```

Press the DO key, type GET FILE OLDFILE.DAT, and press RETURN to create a new buffer containing the text of OLDFILE.DAT in the bottom window of your screen.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                                   Insert      Forward
4 lines read from file WORKDISK:[USER]OLDFILE.DAT
```

Move the cursor to the letter *R* in the word Read, press the Select key, and press the down arrow twice. The last two lines in OLDFILE.DAT are highlighted.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward

Schedule for 1 July
10:00 AM meeting with supervisor
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer OLDFILE.DAT                                   Insert      Forward
Selection started. Press Remove when finished.
```

Press Remove, to place the highlighted text in the Insert Here buffer.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer OLDFILE.DAT                                   Insert      Forward
Remove completed.
```

Press the DO key, type OTHER WINDOW, and press RETURN to move the cursor to the other window. Move the cursor to the bottom of the buffer and press the Insert Here key. The text that you removed from OLDFILE.DAT is inserted into RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
Read and review memo from Donna
Work on Pascal program
[End of file]

Buffer RHYMES.DAT                                    Insert      Forward

Schedule for 1 July
10:00 AM meeting with supervisor
[End of file]

Buffer OLDFILE.DAT                                   Insert      Forward
```

The ONE WINDOW command removes the second window from the screen, expanding the current window to occupy the whole editing area of the screen. Press the DO key, type ONE WINDOW, and press RETURN.

If you exit from the editing session, EVE writes the contents of the current buffer to a file and asks you if you want to write the contents of the other buffer to a file.

3.8 Defining Keys

You can define keys to execute an EVE command or to enter a series of keystrokes called a learn sequence.

EVE does not allow you to define the DO key, the RETURN key (CTRL/M), the space bar, and all printing characters (such as letters, digits, and punctuation marks) on the main keyboard. DIGITAL recommends that you do not define the following keys and control key sequences:

DELETE <X]
F6 (VT200-series)
Help (VT200-series), PF2 (VT100-series)
CTRL/C
CTRL/R
CTRL/S
CTRL/T
CTRL/U
CTRL/Q
CTRL/X
CTRL/Y

You can define all other keys, including control keys.

3.8.1 Defining Keys to Execute an EVE Command

The DEFINE KEY command assigns an EVE command to a single key or control key sequence. You can, in effect, create your own editing keys to enter EVE commands that you use frequently. Key definitions are discarded when you terminate an EVE editing session unless you use the SAVE EXTENDED TPU command (see Section 3.11.2) to save key definitions from one editing session to the next.

To define a key, press the DO key, type DEFINE KEY, and press RETURN. Type the EVE command that you want to assign to the key, press RETURN, and type the key to be associated with the EVE command. The message "Key defined" appears if you have successfully defined a key.

EVE does not provide a simple way to remove a key definition. You can redefine a key, however, by simply associating a different EVE command or learn sequence with the key.

Section 3.8.3 contains an example that defines a key to execute an EVE command.

3.8.2 Defining Keys to Enter a Learn Sequence

The LEARN command assigns a sequence of keystrokes, called a learn sequence, to a single key or control key. Learn sequences allow you to enter the same series of keystrokes in your buffer any number of times, simply by pressing one key. All learn sequences are discarded when you terminate an EVE editing session unless you use the SAVE EXTENDED TPU command (see Section 3.11.2) to save them from one editing session to the next.

To define a learn sequence, press the DO key, type LEARN, and press RETURN. Type the keystrokes to be remembered and press CTRL/R. Press the key to be associated with the learn sequence. The message "Key sequence remembered" appears if you have successfully defined a key.

The following example shows you how to define a learn sequence that is inserted into your file when you press CTRL/F. Invoke EVE to edit the file RHYMES.DAT.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
3 lines read from file WORKDISK:[USER]RHYMES.DAT
```

To begin the definition of the learn sequence, press the DO key, type LEARN, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

Enter the following text in your file.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Press keystrokes to be learned. Press CTRL/R to remember these keystrokes.
```

Press CTRL/R.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Press the key that you want to use to see what was just learned:
```

Press CTRL/F, the key to which you will assign the learn sequence.

```
She rhymes with tree,
also with bee,
and this one makes three.
And what is a rhyme?
[End of file]
Buffer RHYMES.DAT                                    Insert      Forward
Key sequence remembered
```

For the rest of the editing session, simply press CTRL/F and the text "And what is a rhyme?" is inserted at the current cursor position.

3.8.3 Assigning Two Definitions to the Same Key

Use a shift key to assign two definitions (EVE commands or learn sequences) to the same editing key. To define a shift key, press the DO key, type SET SHIFT KEY, and press RETURN. Press the key that you want to use as the shift key. The message "Shift key set" appears in the MESSAGES buffer. Do not confuse this with the keyboard key that you press to capitalize letters. Once you have defined a shift key and have assigned two definitions to an editing key, one editing function is performed by simply pressing the editing key and the other is performed by first pressing the shift key and then pressing the editing key.

The following example defines the number 4 key on the numeric keypad as a shift key, and assigns the BOTTOM and TOP commands to the CTRL/G key. Invoke EVE to edit the file RHYMES.DAT. Press the DO key, type SET SHIFT KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                   Insert      Forward
Press the key that you want to use as the shift key:
```

Press the number 4 key on the numeric keypad.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert        Forward
Shift key set
```

Define the CTRL/G key to enter the BOTTOM command. Press the DO key, type DEFINE KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert        Forward
EVE command:
```

Type BOTTOM, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert        Forward
Press the key that you want to define:
```

Press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert        Forward
Key defined
```

Now, define the shifted CTRL/G key to enter the TOP command. Press the DO key, type DEFINE KEY, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert        Forward
EVE Command:
```

Type TOP, and press RETURN.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert      Forward
Press the key that you want to define:
```

Press the shift key (number 4 on the numeric keypad) and then press CTRL/G.

```
She rhymes with tree,
also with bee,
and this one makes three.
[End of file]
Buffer  RHYMES.DAT                                  Insert      Forward
Key defined
```

For the rest of your editing session, when you press CTRL/G key EVE executes the BOTTOM command; and when you press the shift key (number 4 on the numeric keypad) and CTRL/G, EVE executes the TOP command. Use the SAVE EXTENDED TPU command (see Section 3.11.2) to save key definitions from one editing session to the next.

The results are unpredictable if you define more than one shift key. Use the VAXTPU UNDEFINE_KEY built in to undefine excess shift keys; press the DO key, type TPU UNDEFINE_KEY (key-name), and press RETURN.

3.9 Using DCL with EVE

You can execute a DCL command from within EVE, or you can use subprocesses to switch between DCL command level and EVE editing sessions very quickly.

To execute a DCL command from within EVE, press the DO key, type DCL dcl-command-name, and press RETURN. The message "Creating DCL subprocess . . . " appears in the message buffer. When the DCL command has executed, EVE creates a window and displays the DCL command and its output in the DCL buffer. (The cursor remains in the window it was in before you executed the DCL command.) You can move the cursor to the DCL buffer, select and remove text, and copy it to the editing buffer. DIGITAL recommends that some DCL commands not be issued, such as programs that generate continuous output or programs that do screen management of their own, such as the Phone Utility. The DCL command is most useful when you want to captivate output in a buffer.

You can create subprocesses to switch between an EVE editing session and DCL command level without terminating your editing session. Press the DO key, type SPAWN, and press RETURN. EVE suspends the current editing session and connects your terminal to a new VMS subprocess. The DCL prompt ($) appears on your terminal screen. The most common reasons to spawn a subprocess are to invoke the Mail Utility and to run screen-oriented programs, although your subprocess may invoke any VMS utility or execute any DCL command. To return to your editing session, log out of the subprocess by typing the DCL command LOGOUT and press RETURN. EVE resumes the editing session, and the cursor appears in the location it occupied before you spawned the subprocess.

To keep one EVE editing session active for an entire VMS terminal session, type the DCL command SPAWN and press RETURN. Invoke EVE to edit the file. When you wish to return to DCL command level, press DO, type ATTACH, and press RETURN. Unlike the DCL command ATTACH, which requires a process name as a parameter, the EVE command ATTACH always returns control to the parent process. To return to the EVE editing session, type the DCL command ATTACH with the process name of the EVE subprocess, and press RETURN. EVE resumes the editing session and the cursor appears in the location it occupied before you attached to the parent process.

3.10 Using VAXTPU Commands from Within the EVE Editing Interface

The TPU command allows you to enter any VAXTPU command or series of commands that is one line long on EVE's command line. To enter a VAXTPU command, press the DO key, type TPU tpu-command-name, and press RETURN. For example, to execute the TPU APPEND_LINE command, which places the current line at the end of the previous line, press the DO key, type TPU APPEND_LINE, and press RETURN. The *VAX Text Processing Utility Reference Manual* contains a complete list of VAXTPU commands.

3.11 Extending the EVE Editing Interface to the VAXTPU Text Processing Utility

Because EVE is an editor written in the VAXTPU programming language, you can extend EVE's functions by writing procedures in VAXTPU. This section assumes that you are familiar with the VAXTPU programming language described in the *VAX Text Processing Utility Reference Manual.*

Before you begin writing procedures to modify the EVE editing interface, DIGITAL recommends that you study the EVE source code, which is stored in SYS$SHARE:EVESECINI.TPU. Variables and statements in your procedures should be consistent with EVE's variables and statements so that you can avoid making changes that will negatively affect EVE's operations.

3.11.1 Writing and Compiling VAXTPU Procedures

You can write procedures in the VAXTPU programming language that are, in effect, new EVE commands.

If the EVE command you are writing takes any parameters, such as SET LEFT MARGIN 2, where 2 is the parameter, you must define a global variable to associate that parameter with either a data type string or a data type integer. When you are using the EVE commands that you have defined, EVE passes the null string ("") if you do not provide a value for a string parameter; it passes the value EVE$K_NO_ARG, which is equivalent to a very large negative number, if you do not provide a value for an integer parameter.

DIGITAL recommends placing global variables in a procedure called TPU$LOCAL_INIT, which is called each time EVE starts. Place all procedures that use parameters in the same file that contains the TPU$LOCAL_INIT procedure. Some examples of global variable definitions follow.

```
eve$arg1_add := "integer";
```

This definition of the global variable eve$arg1_add tells EVE to expect an integer as the first parameter for EVE_ADD.

```
eve$arg1_hello := "string";
```

This definition of the global variable eve$arg1_hello tells EVE to expect a string as the first parameter to EVE_HELLO.

In general, each global variable name must consist of the following three parts:

- The first part of the variable name must be **eve$**.
- The second part defines the variable's sequence within a procedure. For the first variable it is **arg1**; for the second, **arg2**; and so on.
- The third part of the variable name is the procedure name without the **EVE_** prefix.

When you write EVE procedures, you should follow these rules:

- Prefix the procedure name with **EVE_** so that EVE recognizes the procedure as an EVE command. After you compile the procedure, execute it by pressing the DO key and typing the procedure name without the EVE_ prefix. (You may also define a key to execute the new command. See Section 3.8.)
- The words **PROCEDURE** and **ENDPROCEDURE** must start in column 1.
- Place all global variable definitions for all EVE commands that use parameters in a single procedure named TPU$LOCAL_INIT.
- Ensure that all identifiers contain a maximum of 132 characters. A VAXTPU identifier is a combination of alphanumeric characters, dollar signs, and underscores that is used to name a parameter, variable, or procedure.
- Ensure that EVE command names contain a maximum of 102 characters, not including the prefix EVE_.

The EXTEND TPU command enables you to compile a VAXTPU procedure without leaving EVE. To compile one procedure, press the DO key, type EXTEND TPU procedure-name and press RETURN. To compile all procedures in a file, press the DO key, type EXTEND TPU * , and press RETURN.

The following example illustrates how to create and compile VAXTPU procedures to define an ADD command, a HELLO command, and the parameters for both commands. Invoke EVE to edit the file MYPROCEDURES.DAT and insert the following text into the file.

```
! Procedure to add two integers and display the result in the message window
PROCEDURE eve_add (a1, a2)
LOCAL temp,
      n1,
      n2;
IF NOT eve$prompt_number(a1, n1, "First number to add: ",
 "No number specified.")
THEN
    RETURN;
ENDIF;
IF NOT eve$prompt_number(a2, n2, "Second number to add: ",
 "No number specified.")
THEN
    RETURN;
ENDIF;
temp := n1 + n2;
MESSAGE (str(n1) + " + " + str(n2) + " = " + STR (temp));
ENDPROCEDURE;
```

```
PROCEDURE eve_hello (my_name)
local the_name;
IF eve$prompt_string(my_name, the_name, "Name: ", "We haven't been introduced")
THEN
    MESSAGE ("Hello " + the_name);
ENDIF;

ENDPROCEDURE;

! Procedure to supply global variables with parameter information
PROCEDURE tpu$local_init
eve$arg1_add := "integer";
eve$arg2_add := "integer";
eve$arg1_hello := "string";
ENDPROCEDURE;
Buffer  MYPROCEDURES.DAT                    Insert      Forward
```

To compile the procedures that you have entered into MYPROCEDURES.DAT, press the DO key, type EXTEND TPU *, and press RETURN. The following section describes how to save the compiled procedure so that you can use the EVE ADD command during an editing session.

3.11.2 Saving VAXTPU Procedures, Key Definitions, and Learn Sequences

The SAVE EXTENDED TPU command saves all currently defined procedures, key definitions, and learn sequences in a section file that you specify. To save the procedures that you have compiled and the key definitions and learn sequences you have created during an editing session, execute the SAVE EXTENDED TPU command before you terminate the editing session. To do so, press the DO key, type the SAVE EXTENDED TPU command in the following format, and press RETURN:

```
Command: SAVE EXTENDED TPU device:[directory]file_name.TPU$SECTION
```

Note that you should include the device, directory, and a file name that you choose. The file type must be TPU$SECTION.

If you specify the same file specification each time you execute the SAVE EXTENDED TPU command, all key definitions, learn sequences, and procedures accumulate in one file from one editing session to the next.

To use this extended version of EVE, you must include the /SECTION qualifier in the command line when you invoke EVE. For example, to invoke EVE to edit the file RHYMES.DAT using the section file WORKDISK:[USER]MYDEFS.TPU$SECTION, use the following command:

```
$ EDIT/TPU/SECTION=WORKDISK:[USER]MYDEFS.TPU$SECTION RHYMES.DAT
```

Remember, you may define a command symbol to invoke this or any other lengthy command line. You can use all the key definitions, learn sequences, and procedures that you previously defined and saved in WORKDISK:[USER]MYDEFS.TPU$SECTION during this editing session.

4 DSR

4.1 What Is DSR?

DIGITAL Standard Runoff (DSR) is a program that formats text. You can use DSR to determine your page size, create justified or unjustified right margins, place space between lines, and form lists. DSR can center titles, supply page numbers, and organize your text into sections and chapters. You can also use DSR to create a table of contents and an index. For a complete list of all the available DSR commands, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

4.1.1 Using DSR Defaults

When you process a file using DSR, a certain set of DSR commands affect your file by default. You do not need to add these DSR commands to your file to achieve the desired effect because DSR does it automatically. If you do not want your file to be affected by these DSR default commands, you must disable them. See the *VAX DIGITAL Standard Runoff (DSR) Reference Manual* for a complete list of the default commands provided by DSR and the commands you need to disable them. The most common of the defaults are discussed here.

When you use DSR to process a file, your output file will look different from your input file because DSR provides the following standard format defaults:

- A text width of 70 characters and a text length of 58 lines.
- Sequential page numbering (omitting a page number for the first page).
- A left margin setting of 0 and a right margin setting of 70.
- Single spacing.
- Tab stops at character positions 9, 17, 25, 33, and so on.
- Filling: DSR fills each line with as many words as possible until the addition of another complete word would exceed the right margin.
- Justification: DSR adds spaces between words to expand each line exactly to the right margin, making the right margin even, or justified.

For example, if you create a file containing the text in the next example, then process the file using DSR, the resulting output file will be 70 characters wide (beginning at character position 0 and ending at character position 70), single spaced, filled, and justified. (The text in the following example is used with permission from the publisher.)

Input File:

```
The use of language begins with imitation.

The infant imitates the sounds made by its
parents;  the child imitates first the spoken
language, then the stuff of books.

The imitative life continues long after the writer
is on his own in the language, for it is almost impossible
to avoid imitating what one admires.
```

Output File:

```
The use of language begins with imitation.  The  infant  imitates  the
sounds  made  by  its  parents;  the  child  imitates first the spoken
language, then the stuff of books.  The imitative life continues  long
after  the  writer  is  on  his  own in the language, for it is almost
impossible to avoid imitating what one admires.
```

4.1.2 Including DSR Commands

To use DSR, you create a file, insert text, and include DSR commands in the appropriate places. DSR creates an output file with a file type of MEM (for example, TEXT.MEM). The MEM file contains the formatted text, which you can then print or display on your terminal.

Use the following steps to format text using DSR:

What You Do	How You Do It
Create a file	$ EDIT list.rno Input file does not exist [EOB] *
Insert text with DSR commands	.literal .LIST .LIST ELEMENT;apples .LIST ELEMENT;plums .LIST ELEMENT;oranges .LIST ELEMENT;bananas .END LIST .end literal
Process file	$ RUNOFF list.rno
Display formatted file	Type LIST.MEM **1** apples **2** plums **3** oranges **4** bananas

4.1.3 Looking at DSR Commands

All DSR commands are English words preceded by a period (.). Here are examples of two DSR commands:

```
.BLANK
.CENTER
```

The .BLANK command inserts a blank line and the .CENTER command centers the text immediately following it on the same line. You can abbreviate all DSR commands. The abbreviations for .BLANK and .CENTER are .B and .C, respectively. For a complete list of valid DSR command abbreviations, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

You end a DSR command with a terminator. Commands are most commonly terminated by the end of a line, but you can also use a semicolon (;) as a terminator, or you can terminate a command and begin another one with a period (.) You can also combine terminators. For example, terminate a command with a semicolon and another command.

The following table shows four ways to terminate a DSR command:

DSR Command	Terminator
#.BLANK2	(end of line)
#.BLANK2;	semicolon
#.BLANK2.CENTER;text	another command
#.BLANK2;.CENTER;text	semicolon and another command

When you add DSR commands to your text, you must place them at the left margin. The following example demonstrates how to use the .BLANK and .CENTER commands to format text in a file called DIETING.RNO:

```
.CENTER;Watching Your Weight Increase ❶
.CENTER;Twelve Days of Dieting
.BLANK3 ❷
On the twelfth  day of dieting, Millitsa gave to me,
.BLANK ❸
Twelve hot fudge sundaes,
.BLANK
Eleven hostess twinkies,
.BLANK
Ten cherry cheese cakes,
.BLANK2 ❹
Nine lady fingers,
.BLANK ❺
Eight date nut muffins,
.BLANK
Seven oatmeal cookies,
.BLANK
Six bags of fritos,
.BLANK2
.CENTER;FIVE COFFEE RINGS, ❻
.BLANK3
Four sticky buns,
.BLANK
Three CLARK bars,
.BLANK
Two marbled cakes,
.BLANK
And a pizza with pepperoni.
```

❶ The .CENTER command terminated by a semicolon (;) centers the title.

❷ The .BLANK command followed by the number 3 creates three blank lines.

❸ The .BLANK command creates one blank line.

❹ The .BLANK command followed by the number 2 creates two blank lines.

❺ The .BLANK command creates one blank line.

❻ The .CENTER command terminated by a semicolon (;) centers the line of text.

When you enter the command RUNOFF DIETING.RNO, DSR will format your text to look like this:

```
                Twelve Days of Dieting

On the twelfth day of dieting, Millitsa gave to me,
Twelve hot fudge sundaes,
Eleven hostess twinkies,
Ten cherry cheese cakes,

Nine lady fingers,
Eight date nut muffins,
Seven oatmeal cookies,
Six bags of fritos,
                  FIVE COFFEE RINGS,

Four sticky buns,
Three CLARK bars,
Two marbled cakes,
And a pizza with pepperoni.
```

4.1.4 Running DSR to Process Your Files

After you add DSR commands to your file and exit from the editor, you are ready to run DSR. To run DSR, you enter the RUNOFF command followed by the name of the file you want to process. For example, to process a file named FUN.FUN, enter the following command line:

```
$ RUNOFF FUN.FUN
```

If you process a file with a file type of RNO, you only need to enter the file name, not the file type. For example, to process a file named FUN.RNO, enter the following the command line:

```
$ RUNOFF FUN
```

4.1.4.1 Using Qualifiers with the RUNOFF Command

DSR provides twenty different qualifiers that you can use with the RUNOFF command. These qualifiers are listed below:

/BACKSPACE
/BOLD
/CHANGE_BARS
/CONTENTS
/DEBUG
/DOWN
/FORM_SIZE
/INDEX
/LOG
/MESSAGES
/NONSPACING_UNDERLINE
/OUTPUT
/PAGES
/PAUSE
/RIGHT
/SEPARATE_UNDERLINE
/SEQUENCE
/SIMULATE
/UNDERLINE_CHARACTER
/VARIANT

Section 4.13 explains how to create a table of contents and an index by using the qualifiers /CONTENTS and /INDEX. All the qualifiers are discussed in detail in the *VAX DIGITAL Standard Runoff (DSR) Reference Manual*. The next section discusses how to use one of the qualifiers, /MESSAGES.

Using the /MESSAGES Qualifier

If DSR finds any errors during processing, it prints an error message (both on the screen and in the MEM file) telling you what the error is and where it is located in both the input and the output files. You can then correct the RNO file (by using a text editor) and reprocess the file.

You can use the /MESSAGES qualifier to specify where you want DSR to display error messages. The options are the following:

OUTPUT	Sends error messages only to the output file (MEM file)
USER	Sends error messages only to the terminal (screen)

The default is /MESSAGES=(OUTPUT,USER), which sends messages to the output file and displays them on the terminal. You can prevent error messages from going either to the output file or to the terminal, but you cannot suppress them entirely.

The following command causes DSR to display error messages in the MEM file, not at the terminal:

```
$ RUNOFF FUN/MESSAGES=OUTPUT
```

The following command causes DSR to display error messages at the terminal, not in the MEM file:

```
$ RUNOFF FUN/MESSAGES=USER
```

4.1.5 Stripping MEM Files of Carriage-Return/Line-Feed Symbols

When you edit a MEM file produced by DSR, you will notice that the file contains carriage-return/line-feed symbols (<CR> <LF>). You can also have a file containing these symbols when you add a MEM file to an RNO file. In either case, you may want to strip the MEM file of these symbols.

The following example shows an input file (named MESS.RNO), the processed file (MESS.MEM), and the way this same MEM file looks with the <CR> <LF> 's when you invoke EDT to edit it (MESS.MEM):

Input file (MESS.RNO):

```
This is the first line of a file named MESS.RNO.
.BLANK 2
If you try to edit this file's MEM file after it is processed,
you will see 12 <CR><LF>'s.
.BLANK 2
There is a way to strip the MEM file of these confusing symbols.
```

Output file (MESS.MEM):

```
This is the first line of a file named MESS.RNO.

If you try to edit this file's MEM file after it is processed,
you will see 12 <CR><LF>'s.

There is a way to strip the  MEM  file  of  these  confusing
symbols.
```

If you try to edit the output file (MESS.MEM) with EDT, you see the following:

```
<CR><LF>
<CR><LF>
<CR><LF>
This is the first line of a file named MESS.RNO.<CR><LF>
<CR><LF>
<CR><LF>
If you try to edit this file's MEM file after it is processed,<CR><LF>
you will see 12 <CR><LF>'s.<CR><LF>
<CR><LF>
<CR><LF>
There is a way to strip the  MEM  file  of  these  confusing<CR><LF>
symbols.<CR><LF>
```

However, you can use the TECO editor to strip the MEM file of the unwanted <CR> <LF> 's:

```
$ TECO mess.mem
*EX<ESC><ESC>
$
```

4.2 How to Format Lists

You need to know only the following three DSR commands to format a numbered list:

1 _.LIST

2 _.LIST ELEMENT

3 _.END LIST

Begin your list with the .LIST command. The .LIST command causes DSR to start a list. When DSR formats a list, it indents the left margin, puts a blank line before the first list item, after the last item, and between each item in the list. DSR also numbers (or otherwise marks) the items.

Precede each item in your list by the .LIST ELEMENT command. Use a semicolon to separate the list item from the command or put the list item on the following line.

End your list with the .END LIST command. The following example shows how to format a list using DSR commands:

Input file (LIST.RNO):

```
.LIST
.LIST ELEMENT;grosbeak
.LIST ELEMENT;gold finch
.LIST ELEMENT;redpoll
.LIST ELEMENT;sparrow
.END LIST
```

Output file (LIST.MEM)

```
1.  grosbeak
2.  gold finch
3.  redpoll
4.  sparrow
```

The following sections describe other kinds of lists and the DSR commands you need to create them.

4.2.1 Creating Bulleted Lists

By default, DSR creates a numbered list. If you do not want your list to be numbered, you can substitute bullets or other characters for the numbers. For example, if you want a bulleted list, you enter the .LIST command followed by a lowercase o, or a bullet in quotation marks ("o"):

```
.LIST "o"
```

The following example shows how to make a bulleted list:

Input file (LIST.RNO):

```
.LIST "o"
.LIST ELEMENT;ferret
.LIST ELEMENT;mink
.LIST ELEMENT;rabbit
.LIST ELEMENT;sable
.LIST ELEMENT;raccoon
.END LIST
```

Output file (LIST.MEM):

- ferret
- mink
- rabbit
- sable
- raccoon

4.2.2 Creating Lists Using Any Symbol

If you do not want numbered or bulleted lists, you can choose whatever character or symbol you want and enclose it in quotation marks following the .LIST command. Asterisks (*), dollar signs ($), or at signs (@) are used in the following examples:

Input file (LIST.RNO):

```
.LIST "*"
.LIST ELEMENT;jupiter
.LIST ELEMENT;mars
.LIST ELEMENT;venus
.END LIST
```

Output file (LIST.MEM):

* jupiter
* mars
* venus

Input file (LIST.RNO):

```
.LIST "$"
.LIST ELEMENT;mark
.LIST ELEMENT;yen
.LIST ELEMENT;buck
.LIST ELEMENT;pound
.END LIST
```

Output file (LIST.MEM):

```
$ mark
$ yen
$ buck
$ pound
```

Input file (LIST.RNO):

```
.LIST "@"
.LIST ELEMENT;why
.LIST ELEMENT;not
.LIST ELEMENT;?
.END LIST
```

Output file (LIST.MEM):

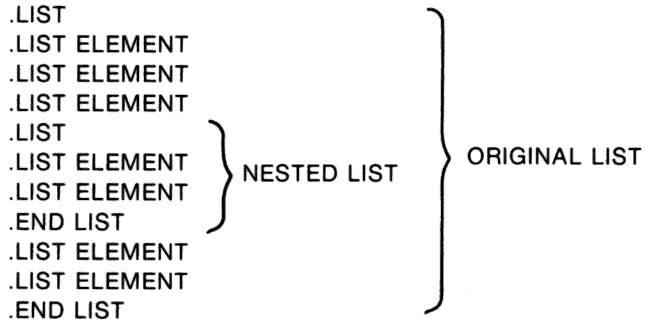
```
@ why
@ not
@ ?
```

4.2.3 Creating Nested Lists

You can also make a nested list—that is, a list indented within a list. You simply issue another .LIST command between the original .LIST and .END LIST commands. The new .LIST command temporarily suspends the characteristics of the original list. The next .END LIST command ends only the new .LIST and restores the characteristics of the previous one. You must terminate each list with the .END LIST command.

Figure 4–1 displays the original list containing five elements and the nested list containing three elements.

Figure 4–1 Creating a Nested List



ZK-1264-83

Notice that the nested list in the following example has bullets unlike the original numbered list:

Input file (LIST.RNO):

```
.LIST
.LIST ELEMENT;German
.LIST ELEMENT;Russian
.LIST ELEMENT;Swedish
.LIST ELEMENT;Yugoslavian
.LIST "o"
.LIST ELEMENT;Serbian
.LIST ELEMENT;Croatian
.LIST ELEMENT;Macedonian
.END LIST
.LIST ELEMENT;Turkish
.LIST ELEMENT;Scottish
.LIST ELEMENT;Irish
.END LIST
```

Output file (LIST.MEM):

```
1. German
2. Russian
3. Swedish
4. Yugoslavian
     o Serbian
     o Croatian
     o Macedonian
5. Turkish
6. Scottish
7. Irish
```

4.2.4 Creating Lists with Letters and Roman Numerals

By default, DSR numbers lists with decimal numbers. To change the default decimal numbers to letters or roman numerals, use the .DISPLAY ELEMENTS command. Use the following syntax:

```
.DISPLAY ELEMENTS x
```

The argument x is a one- or two-letter code that specifies the form that the list numbers will take. The codes and their explanations are shown in the following table:

Code	Form of Sequence and Case
D	Decimal Numbers
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
LU	Uppercase Letters
LL	Lowercase Letters

The following examples show lowercase roman numerals and letters. Notice that the .DISPLAY ELEMENTS command appears between the .LIST command and the first .LIST ELEMENT command.

Input file (LIST.RNO):

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;tan
.LIST ELEMENT;beige
.LIST ELEMENT;rust
.LIST ELEMENT;brown
.END LIST
```

Output file (LIST.MEM):

```
  i.  tan
 ii.  beige
iii.  rust
 iv.  brown
```

Input file (LIST.RNO):

```
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;january
.LIST ELEMENT;february
.LIST ELEMENT;march
.END LIST
```

Output file (LIST.MEM):

```
  a.  january
  b.  february
  c.  march
```

In the following example, notice that each .END LIST command ends the sequence that you specified with .DISPLAY ELEMENTS. You must issue the .DISPLAY ELEMENTS command each time you start a list in which you want to change the numbers or letters. If you want each list to be marked by the same characters, you must still issue a .DISPLAY ELEMENTS command for every .LIST command.

Input file (LIST.RNO):

```
.LIST
.DISPLAY ELEMENTS RL
.LIST ELEMENT;marble
.LIST ELEMENT;maple
.LIST ELEMENT;alabaster
.LIST ELEMENT;oak
.LIST
.DISPLAY ELEMENTS LL
.LIST ELEMENT;light
.LIST ELEMENT;dark
.END LIST
.LIST ELEMENT;glass
.LIST ELEMENT;plastic
.LIST ELEMENT;mahogany
.END LIST
```

Output file (LIST.MEM):

```
  i.  marble
 ii.  maple
iii.  alabaster
 iv.  oak
      a.  light
      b.  dark
  v.  glass
 vi.  plastic
vii.  mahogany
```

4.3 How to Format Memos

You can use the following DSR commands to format a memo:

1 _.BLANK

2 _.BREAK

3 _.TAB STOPS

You can organize these three DSR commands into a memo skeleton, like the following one, and fill in the variable information every time you need to send a memo. (Notice that you must press the TAB key to activate the .TAB STOPS command.)

Input file (SKELETON.RNO):

```
.TAB STOPS 30
<TAB>   DATE:
.BREAK
<TAB>   FROM:
.BREAK
<TAB>   DEPT:
.BREAK
<TAB>   EXT:
.BREAK
<TAB>   LOC:
.BLANK 2
TO:
.BLANK
SUBJECT:
.BLANK 2
(Enter text of memo here.)
```

Output file (SKELETON.MEM):

```
                                DATE:
                                FROM:
                                DEPT:
                                EXT:
                                LOC:

TO:

SUBJECT:

(Enter text of memo here.)
```

The .BREAK command ends the current line immediately, without filling or justification. The .TAB STOPS n command changes the current position of the tab stops. The value of n specifies the character position for a tab stop.

The following memo example uses the following DSR commands:

```
.BLANK
.BREAK
.LEFT MARGIN
.LIST
.LIST ELEMENT
.END LIST
.LITERAL
.END LITERAL
.TAB STOPS
```

Input file (SPY.RNO):

```
❶       .TAB STOP 40
              <TAB>DATE: 13-Nov-85
❷       .BREAK
               <TAB>FROM: A. Gent
               .BREAK
               <TAB>DEPT: Topspy
               .BREAK
               <TAB>EXT: 007
               .BREAK
               <TAB>LOC: Swiss Branch
               .BLANK 2
               TO: Underlings
               .BLANK 2
               SUBJECT: New Equipment
               .BLANK 2
               The following items will be discussed in our
               next meeting:
❸       .LIST
               .LIST ELEMENT;super slim tie tack monitor
               .LIST ELEMENT;wired socks
               .LIST ELEMENT;mini diamond stud earring camera
               .END LIST
```

```
                Please attempt to decode the following message
                for your next assignment:
❹          .LEFT MARGIN 20
❺          .LITERAL
                                que estap dyi
                                    fho
                                         syto
                               szru
                                     fsohg
                                    wquip
                                  dwd
❻          .END LITERAL
❼          .LEFT MARGIN 0
```

❶ The .TAB STOPS 40 command sets the first tab stop to character position 40.

❷ The .BREAK command ends the current line immediately and starts a new line without adding a blank line.

❸ The .LIST command begins a list.

❹ The .LEFT MARGIN 20 command sets the left margin to 20.

❺ The .LITERAL command displays the text literally, exactly as it is entered.

❻ The .END LITERAL command stops displaying the text literally.

❼ The .LEFT MARGIN 0 command sets the left margin back to 0.

Output file (SPY.MEM):

```
                                        DATE: 13-Nov-85
                                        FROM: A.  Gent
                                        DEPT: Topspy
                                        EXT: 007
                                        LOC: Swiss Branch

TO: Underlings

SUBJECT: New Equipment

The following items will be discussed in our next meeting:
     1.  super slim tie tack monitor
     2.  wired socks
     3.  mini diamond stud earring camera
```

```
Please attempt to decode the following message for your next
assignment:

                que estap dyi
                    fho
                         syto
              szru
                      fsohg
                     wquip
                  dwd
```

4.4 How to Fill and Justify Text

By default, DSR fills and justifies text (.FILL and .JUSTIFY commands). When you process a file, these commands take effect automatically. The .FILL command causes DSR to fill each line with as many words as possible until the addition of another word would exceed the right margin. The .JUSTIFY command causes DSR to add spaces between words to expand each line exactly to the right margin, making the margin even, or justified. If you do not want DSR to fill or justify the text in your file, you must disable these commands by using the .NO FILL and .NO JUSTIFY commands.

The following example demonstrates how DSR fills and justifies text by default:

DSR commands: .FILL and .JUSTIFY (defaults)

```
Input file (VERSE.RNO):

For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.

Output file (VERSE.MEM):

For it so falls out, That what  we  have
we prize not to the worth while we enjoy
it; but being lacked, lacked  and  lost,
Why, then we rack the value.
```

When you do not want each line of your text to fill with words, use the .NO FILL command. Using the .NO FILL command with the .JUSTIFY command, produces the following variation of the original example. Notice how the .JUSTIFY command spaces the words on each line to create an even right margin.

DSR commands: .NO FILL and JUSTIFY

```
Input file (VERSE.RNO):

.NO FILL
.JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.

Output file (VERSE.MEM):

For      it      so      falls      out,
That      what      we       have
we          prize          not
to          the          worth
while        we        enjoy         it;
but        being        lacked,
lacked        and         lost,
Why,
then     we     rack     the      value.
```

When you want each line of text to fill with words, but do not want an even right margin, use the .FILL and .NO JUSTIFY commands. Notice the ragged right margins in the following example:

DSR commands: .FILL and .NO JUSTIFY

```
Input file (VERSE.RNO):

.FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.

Output file (VERSE.MEM):

For it so falls out, That what we have
we prize not to the worth while we enjoy
it; but being lacked, lacked and lost,
Why, then we rack the value.
```

When you want your text to stay the way you type it, that is, without filling and justification, use the .NO FILL and .NO JUSTIFY commands.

DSR commands: .NO FILL and .NO JUSTIFY

```
Input file (VERSE.RNO):
.NO FILL
.NO JUSTIFY
.RIGHT MARGIN 40
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
Output file (VERSE.MEM):
For it so falls out,
That what we have
we prize not
to the worth
while we enjoy it;
but being lacked,
lacked and lost,
Why,
then we rack the value.
```

(The text used for these examples is an excerpt from William Shakespeare's "Much Ado About Nothing.")

Reminder

DSR will fill and justify text by default unless you specify otherwise with the .NO FILL and .NO JUSTIFY commands.

4.5 How to Adjust the Display of Text

By default, DSR provides a page length of 58 lines and a page width of 70 characters. When you want to change the size of your page, use the .PAGE SIZE and .RIGHT MARGIN commands. The syntax follows:

```
.PAGE SIZE n
.RIGHT MARGIN n
```

The parameter n indicates the length. The length is the maximum number of lines on a page. It cannot be smaller than 13. You can control the width of a line with the .RIGHT MARGIN command. N is the maximum number of characters on a line. It cannot be larger than 150.

The following example shows how to create two different page sizes by using the .PAGE SIZE command with the .RIGHT MARGIN command:

Input file (PAGESIZE.RNO):

```
.PAGE SIZE 15
.RIGHT MARGIN 20
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
page size page size 15 lines long 20 characters wide
```

Output file (PAGESIZE.MEM):

```
page size page  size
15   lines  long  20
characters wide page
size  page  size  15
lines    long     20
characters wide page
size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide page
                Page 2

size  page  size  15
lines     long    20
characters wide page
size  page  size  15
lines    long     20
characters wide
```

Input file (PAGE.RNO):

```
.PAGE SIZE 25
.RIGHT MARGIN 35
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
page size page size 25 lines long 35 characters wide
```

Output file (PAGE.MEM):

```
page size page size 25 lines long
35 characters wide page size page
size 25 lines long 35 characters
wide page size page size 25 lines
long 35 characters wide page size
page size 25 lines long 35
characters wide page size page size
25 lines long 35 characters wide
page size page size 25 lines long
35 characters wide page size page
size 25 lines long 35 characters
wide page size page size 25 lines
long 35 characters wide page size
page size 25 lines long 35
characters wide page size page size
25 lines long 35 characters wide
page size page size 25 lines long
35 characters wide page size page
size 25 lines long 35 characters
wide page size page size 25 lines
long 35 characters wide page size
page size 25 lines long 35
                                Page 2
characters wide page size page size
25 lines long 35 characters wide
page size page size 25 lines long
35 characters wide page size page
size 25 lines long 35 characters
wide
```

4.5.1 Indenting Text

By default, when you use the .INDENT command, DSR indents the first line of text following the command by five spaces. The syntax for the indent command follows:

```
.INDENT n
```

"N" specifies the number of character positions to the right of the .LEFT MARGIN setting the line of text is indented. When you specify a negative number (–n), DSR moves the line of text n number of character positions to the left of the .LEFT MARGIN setting.

The following example demonstrates how to use the .INDENT command:

Input file (INDENT.RNO):

```
.LEFT MARGIN 10
.RIGHT MARGIN 60
The left margin setting for this text is 10 and the right margin
setting is 60.  If you want to indent text, use the .INDENT command.
```

```
.INDENT 20
This line of text is indented 20 spaces.
Every line you type from now on will be flush with the left
margin until you enter the .INDENT command again.
.INDENT 10
This line of text is indented 10 spaces.
If you want a line of text to begin to the left of the
left margin setting, specify a negative number (-n) with
the .INDENT command.
.INDENT -10
This line of text is 10 spaces to the left of the left margin.
This line of text begins at the left margin setting.  Any text
you enter at this point will remain flush with the left
margin until you enter the .INDENT command again.  If you
want to indent two or three lines of text, you must enter
the .INDENT command on each line you want to indent.
.INDENT 15
This is the first of three indented lines.
.INDENT 15
This is the second of three indented lines.
.INDENT 15
This is the third.
This line of text is back at the left margin again.
```

Output file (INDENT.MEM):

```
          The left margin setting for this text  is  10  and
          the  right  margin  setting is 60.  If you want to
          indent text, use the .INDENT command.

                                This line of text is  indented
          20  spaces.   Every line you type from now on will
          be flush with the left margin until you enter  the
          .INDENT command again.

                    This line of text is indented 10 spaces.
          If you want a line of text to begin to the left of
          the left margin setting, specify a negative number
          (-n) with the .INDENT command.

This line of text is 10 spaces  to  the  left  of  the  left
          margin.   This  line  of  text  begins at the left
          margin setting.  Any text you enter at this  point
          will  remain  flush with the left margin until you
          enter the .INDENT command again.  If you  want  to
          indent  two or three lines of text, you must enter
          the .INDENT command  on  each  line  you  want  to
          indent.

                         This is the first of three indented
          lines.

                         This  is  the   second   of   three
          indented lines.

                         This is the third.  This  line  of
          text is back at the left margin again.
```

4.5.2 Placing a Single Line of Text Relative to the Right Margin

You can use the .RIGHT command to position a single line of text relative to the right margin. The syntax follows:

```
.RIGHT n;text
```

The number "n" can be either positive or negative. A positive number specifies how many character positions to the left of the right margin setting the line will be indented. A negative number specifies the number of character positions to the right of the right margin setting that the line will extend to. "Text" indicates the text to be positioned relative to the right margin. No other DSR command can follow this text on a line.

The following example shows the .RIGHT command being used to notate various directions in the script of a play. (A positive number is specified for "n.")

Input file (PLAY.RNO):

```
.LEFT MARGIN 0
.RIGHT MARGIN 50
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT 25;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT 25;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT 25;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT 25;(open door)
```

Output file (PLAY.MEM):

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                    (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
              (shake head)
all you can think about is yourself.
```

```
Celia:  Maybe I could apologize...
                        (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
                  (open door)
```

The following example is identical to the previous example, except that a negative number is specified for "n."

Input file (PLAY.RNO):

```
Celia: What did you want me to do?  I really tried to
understand what was going on, but I was scared!
.RIGHT -5;(sob)
.BLANK 1
Bert: Maybe some sensitivity would help?  You never
think of anyone but yourself!  Did you ever stop to
consider that maybe he was just as scared as you were?
No,
.RIGHT -5;(shake head)
all you can think about is yourself.
.BLANK 1
Celia: Maybe I could apologize...
.RIGHT -5;(turn)
make amends...
.BLANK 1
Bert: You had better figure out something because it
is your problem now.
.RIGHT -5;(open door)
```

Output file (PLAY.MEM):

```
Celia:  What did you want  me  to  do?   I  really
tried  to  understand what was going on, but I was
scared!
                                                (sob)
Bert:  Maybe some  sensitivity  would  help?   You
never  think of anyone but yourself!  Did you ever
stop to consider that maybe he was just as  scared
as you were?  No,
                                          (shake head)
all you can think about is yourself.
Celia:  Maybe I could apologize...
                                                (turn)
make amends...
Bert:  You had better figure out something because
it is your problem now.
```

4.6 How to Create Space on Your Page

If you want an example or figure to accompany your text, you will need to create some space on the page for it. You can use several different DSR commands to create that necessary space. Some of these commands are:

1 _.BLANK

2 _.FIGURE

3 _.FIGURE DEFERRED

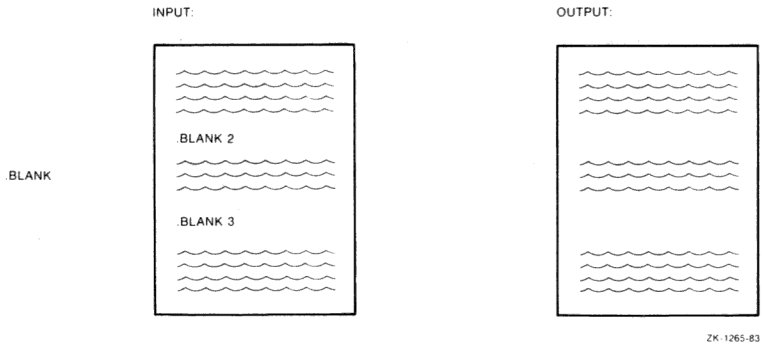
4 _.LITERAL

4.6.1 Separating Sections with Blank Lines

If you use the .BLANK command, you will get the specified number of lines left blank. For example, .BLANK 2 produces two blank lines and .BLANK 10 produces ten blank lines. The .BLANK command is useful for creating one or two blank lines between sections of text. (.BLANK will not work correctly when you use it at the top of a page.) If you issue the .BLANK 10 command near the bottom of the page, there may not be enough room on the page for all 10 lines, resulting in your space being split. If you want to avoid possible space splits, use either the .FIGURE command or the .FIGURE DEFERRED command.

Figure 4–2 shows an input file (TEXT.RNO) containing the .BLANK command and the resulting output file (TEXT.MEM) with the blank line.

Figure 4–2 Using the .BLANK Command



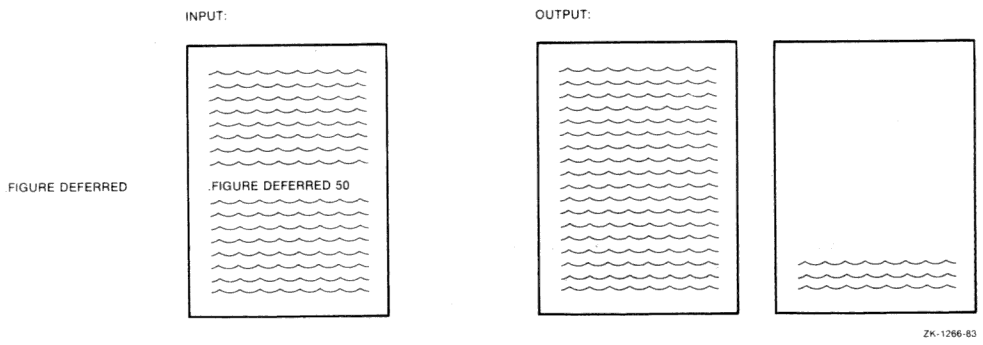
4.6.2 Creating Uninterrupted Space

The .FIGURE DEFERRED command will leave enough space on the current page for the specified number of lines. If all the lines will not fit, .FIGURE DEFERRED will fill the current page with text and create the space for the specified number of lines at the top of the next page.

For example, if you have a figure that is 40 lines long and you issue the command .FIGURE DEFERRED 40, but there are only 25 lines lines left on the page, what happens? The .FIGURE DEFERRED command causes the current page to fill with text and the next page to have a 40-line space at the top for your figure.

Figure 4–3 shows an input file (TEXT.RNO) containing the .FIGURE DEFERRED command and the resulting output file (TEXT.MEM) with the blank lines.

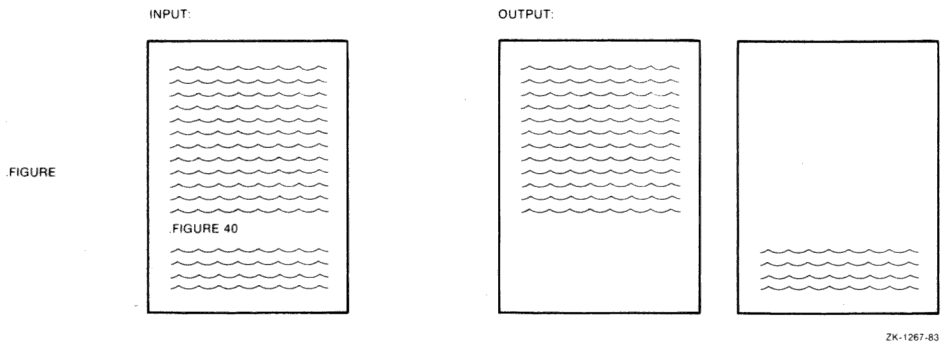
Figure 4–3 Using the .FIGURE DEFERRED Command



If you use the .FIGURE command, the next page will also have a 40-line space at the top, but the current page will end immediately.

Figure 4–4 shows an input file (TEXT.RNO) containing the .FIGURE command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 4–4 Using the .FIGURE Command

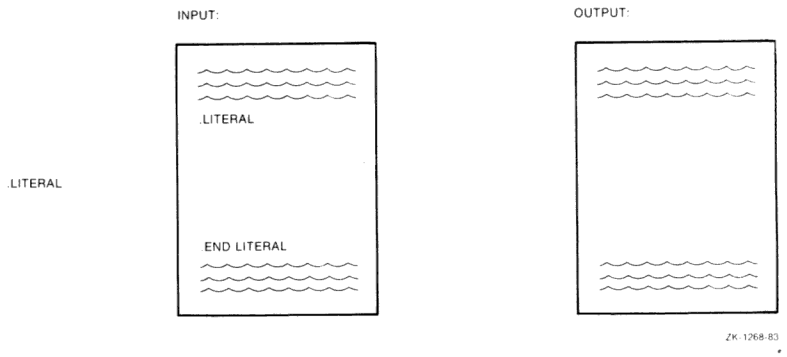


4.6.3 Seeing the Space You Create

You can also use the .LITERAL command to create space. If you want to see the amount of space you are creating, use the .LITERAL command and press RETURN until the screen has enough blank space. Then, issue the .END LITERAL command. You may want to use the .LITERAL command if you have a picture to paste directly on to your text and want to see how it will look with the blank space you create.

Figure 4–5 shows an input file (TEXT.RNO) containing the .LITERAL command and the resulting output file (TEXT.MEM) with the blank lines.

Figure 4–5 Using the .LITERAL Command



The following example shows how to create space using the .BLANK, .FIGURE DEFERRED, .FIGURE, and .LITERAL commands.

Input file (space.rno):

```
.page size 20,40
This is the first line on the page.
Each page will be 20 lines long and 40
characters wide.  If you enter
the .BLANK 2 command here, DSR
creates two blank lines.
.BLANK 2
There is not enough space on this page
for a figure that is 15 lines long, so
if you enter the .FIGURE DEFERRED 15
command here,
.FIGURE DEFERRED 15
DSR will fill up the rest of this page
with text, then create 15 lines of
space at the top of the next page.
texttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,
.FIGURE 10
```

```
DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
But, if you enter the .FIGURE 10 command again,
and there is not enough space left on this
page for the entire figure (in this case 10),
DSR stops displaying text immediately
.FIGURE 10
and moves to the top of the next page, creating
the specified space (10 lines).  Then DSR
resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and press
<RET> three times,
.LITERAL

.END LITERAL
you will create three blank lines.  Remember to
enter the .END LITERAL command when you are
done.
```

Output file (space.mem):

```
This is the first line on the page.
Each page will be 20 lines long and 40
characters wide.  If you enter the
.BLANK 2 command here, DSR creates two
blank lines.

There is not enough space on this page
for a figure that is 15 lines long, so
if you enter the .FIGURE DEFERRED 15
command here,
DSR will fill up the rest of this page
with text, then create 15 lines of space
at the top of the next page.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
                                    Page 2


texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
```

Page 3

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .FIGURE 10 command here,

DSR will create 10 lines of space.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext

Page 4

texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
But, if you enter the .FIGURE 10 command again, and there is not enough space left on this page for the entire figure (in this case 10), DSR stops displaying text immediately

Page 5

and moves to the top of the next page, creating the specified space (10 lines). Then DSR resumes the display of text.
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
texttexttexttexttexttexttexttexttexttext
If you enter the .LITERAL command and

Page 6

press <RET> three times,

you will create three blank lines. Remember to enter the .END LITERAL command when you are done.

4.7 How to Format Sections

When you want to organize text into sections, use the .HEADER LEVEL command. The syntax for the command is:

```
.HEADER LEVEL n title
```

The value of "n" indicates the specific header level. For example, .HEADER LEVEL 1 creates a section beginning with 1. If you enter a series of .HEADER LEVEL 1 commands, DSR produces section numbers starting with 1 and increasing by 1. For example:

Input file (HEAD1.RNO):

```
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
.HEADER LEVEL 1
```

Output file (HEAD1.MEM):

```
1

2

3

4

5
```

The .HEADER LEVEL 2 command affects the second digit. For example, if you enter a series of .HEADER LEVEL 2 commands, you will see the second digit change.

Input file (HEAD2.RNO):

```
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
.HEADER LEVEL 2
```

Output file (HEAD2.MEM):

```
0.1

0.2

0.3

0.4
```

The .HEADER LEVEL 3 command affects the third digit. An example of a series of .HEADER LEVEL 3 commands follows:

Input file (HEAD3.RNO):

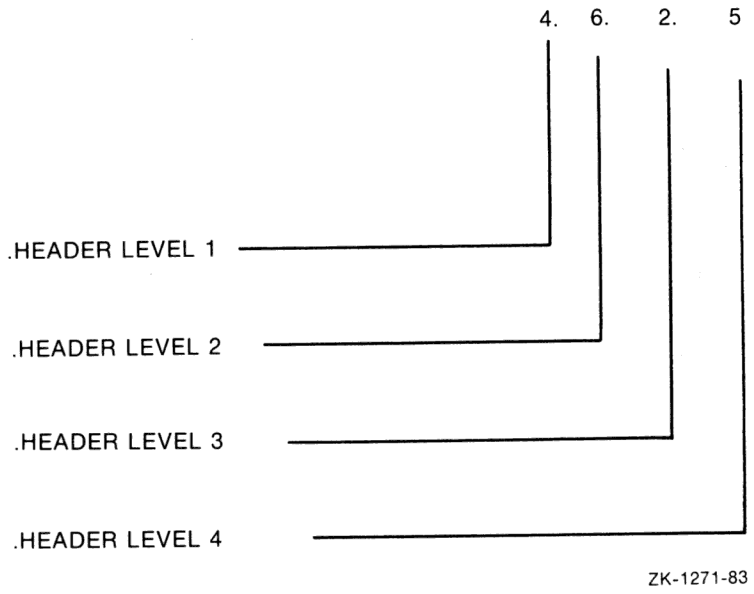
```
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
.HEADER LEVEL 3
```

Output file (HEAD3.MEM):

```
0.0.1
0.0.2
0.0.3
0.0.4
```

Figure 4–6 shows one section number (4.6.2.5) and the various header levels corresponding to each part of the number.

Figure 4–6 Looking at Header Levels



4.7.1 Specifying a Title

You can follow the .HEADER LEVEL n command with a title. DSR uses uppercase for all the letters in a title of header level 1. A title of header level 2 has only initial caps. Header level 3 titles have initial caps followed by a hyphen to separate the text that is run in on the same line as the title.

The following example contains a series of .HEADER LEVEL commands:

Input file (level.rno):

```
.HEADER LEVEL 1this is a header level 1 title
.HEADER LEVEL 1this is another header level 1 title
.HEADER LEVEL 1this is the third header level 1 title in a row
.HEADER LEVEL 2this is a header level 2 title
.HEADER LEVEL 2this is another header level 2 title
.HEADER LEVEL 3this is a header level 3 title
This is the first line of text.
.HEADER LEVEL 1this is the fourth header level 1 title
```

Output file (level.mem):

```
1  THIS IS A HEADER LEVEL 1 TITLE

2  THIS IS ANOTHER HEADER LEVEL 1 TITLE

3  THIS IS THE THIRD HEADER LEVEL 1 TITLE IN A ROW

3.1  This Is A Header Level 2 Title

3.2  This Is Another Header Level 2 Title

3.2.1  This Is A Header Level 3 Title - This is the first line of text.

4  THIS IS THE FOURTH HEADER LEVEL 1 TITLE
```

The following example demonstrates how to organize recipes using the .HEADER LEVEL command:

Input file (RECIPES.RNO):

```
.HEADER LEVEL 1pies
An introduction to pies goes here.
.HEADER LEVEL 2grasshopper pie
The origin of the Grasshopper Pie goes here.
.HEADER LEVEL3chocolate wafer crust
This section explains how to make the crust.
.HEADER LEVEL3filling
This section explains how to make the filling.
.HEADER LEVEL2lemon meringue pie
Various lemon pies are discussed here.
.HEADER LEVEL3flaky pie crust
The recipe for a good, flaky crust goes here.
.HEADER LEVEL3lemon filling
This section describes how to make the filling.
.HEADER LEVEL3meringue topping
Meringues are mentioned here.
.HEADER LEVEL1layered cakes
Layered cakes are described here.
.HEADER LEVEL2millie's magnificent torte
Tortes are described here.
.HEADER LEVEL3walnut layers
A recipe for the walnut layers goes here.
.HEADER LEVEL3coffee frosting
Coffee frosting is explained here.
```

Output file (RECIPES.MEM):

```
1  PIES
An introduction to pies goes here.

1.1  Grasshopper Pie
The origin of the Grasshopper Pie goes here.

1.1.1  Chocolate Wafer Crust - This section explains how  to
make the crust.

1.1.2  Filling - This  section  explains  how  to  make  the
filling.

1.2  Lemon Meringue Pie
Various lemon pies are discussed here.

1.2.1  Flaky Pie Crust - The recipe for a good, flaky  crust
goes here.

1.2.2  Lemon Filling - This section describes  how  to  make
the filling.

1.2.3  Meringue Topping - Meringues are mentioned here.

2  LAYERED CAKES
Layered cakes are described here.

2.2  Millie's Magnificent Torte
Tortes are described here.

2.2.1  Walnut Layers - A recipe for the walnut  layers  goes
here.

2.2.2  Coffee Frosting - Coffee frosting is explained here.
```

4.7.2 Using Roman Numerals or Letters

By default, DSR displays decimal numbers when you use the .HEADER LEVEL command. If you want to specify roman numerals or letters, use the .DISPLAY LEVELS command. The syntax for this command follows:

```
.DISPLAY LEVELS y1,y2,y3...y6
```

The letter "y" is a one- or two-letter code. Several available codes follow:

Code	Form of Sequence and Case
RU	Roman Uppercase Numerals
RL	Roman Lowercase Numerals
LU	Letters, Uppercase
LL	Letters, Lowercase

For more information and a complete list of available codes, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

If you want the numbers in the first level heads to be uppercase roman numerals, the second level to be uppercase letters, and the third level to be lowercase roman numerals, enter the .DISPLAY LEVELS RU,LU,RL command. The resulting sections will look like this:

```
I  ALL HEADER LEVEL 1 TITLES ARE UPPERCASED

The .HEADER LEVEL 1 command produced this title.  The code
RU tells DSR to make all level 1 heads uppercase roman
numerals.

I.A  Header Level 2 Titles Have Initial Uppercase Letters

The .HEADER LEVEL 2 command produced this title.  The code
LU tells DSR to make all level 2 heads uppercase letters.

I.A.i  Header Level 3 Titles Are Followed By A  Hyphen - The
.HEADER LEVEL 3 command produced this title.  The code RL
tells DSR to make all level 3 heads lowercase roman
numerals.
```

4.8 How to Format Chapters

When you want to organize your text into chapters, use the .CHAPTER command. The .CHAPTER command specifies the beginning of a chapter, numbers it, and allows you to supply a title. The syntax for the command is:

```
.CHAPTER title
```

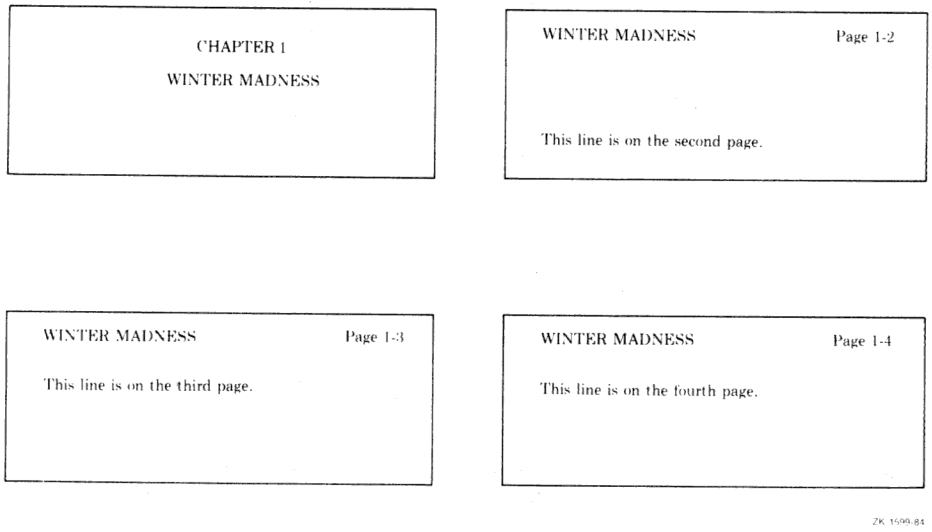
In the following example, the chapter title is "Winter Madness:"

Input file (CHAP.RNO):

```
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 15
This line is on the third page.
.BLANK 22
This line is on the fourth page.
```

Figure 4–7 shows the output file (CHAP.MEM).

Figure 4–7 Using the .CHAPTER Command



4.8.1 Using Roman Numerals or Letters

By default, the .CHAPTER command produces decimal chapter numbers. If you want roman numerals or letters instead of decimal numbers, use the .DISPLAY CHAPTER command. The syntax for this command follows:

```
.DISPLAY CHAPTER y
```

The letter "y" is one of the codes discussed in Section 4.7.2.

For more information about all the available codes, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

You must first issue the .DISPLAY CHAPTER command and then the .CHAPTER command.

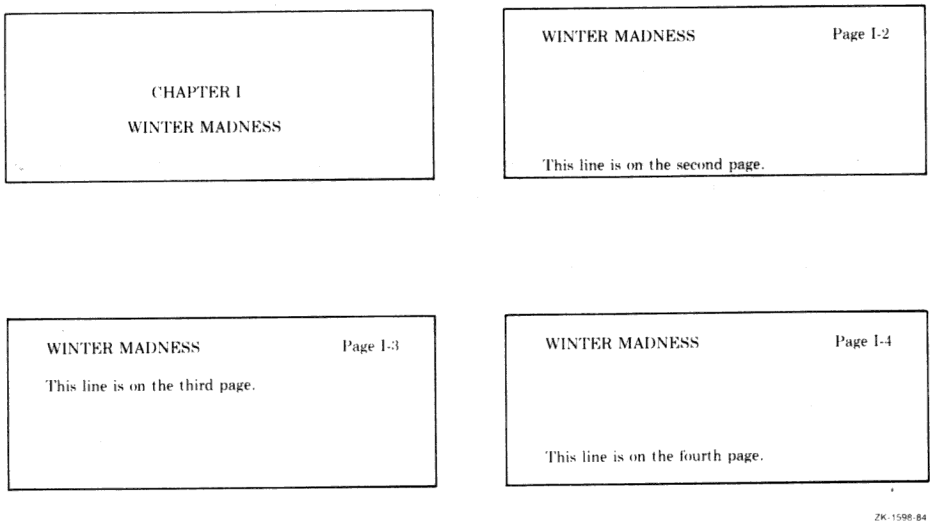
The following example shows how to create chapters numbered with uppercase roman numerals by using the .DISPLAY CHAPTER command with the code RU. Notice the uppercase roman numeral page numbers.

Input file (CHAP.RNO):

```
.DISPLAY CHAPTER RU
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 15
This line is on the third page.
.BLANK 22
This line is on the fourth page.
```

Figure 4–8 shows the output file (CHAP.MEM).

Figure 4–8 Using the .DISPLAY CHAPTER Command



4.8.2 Changing the Way Pages Are Numbered

By default, DSR numbers pages using decimal numbers. If you want your pages lettered or numbered with roman numerals, use the .DISPLAY NUMBER command. The syntax for this command follows:

```
.DISPLAY NUMBER y
```

The letter "y" is one of the codes discussed in Section 4.7.2.

The following example shows the commands you use to specify chapter and page numbers.

Input file (CHAP.RNO):

```
.DISPLAY CHAPTER RU ❶
.CHAPTER winter madness
.PAGE SIZE 20
.BLANK 15
This line is on the second page.
.BLANK 15
7DISPLAY NUMBER LL ❷
This line is on the third page.
.BLANK 30
.DISPLAY NUMBER RL ❸
This line is on the fourth page.
```

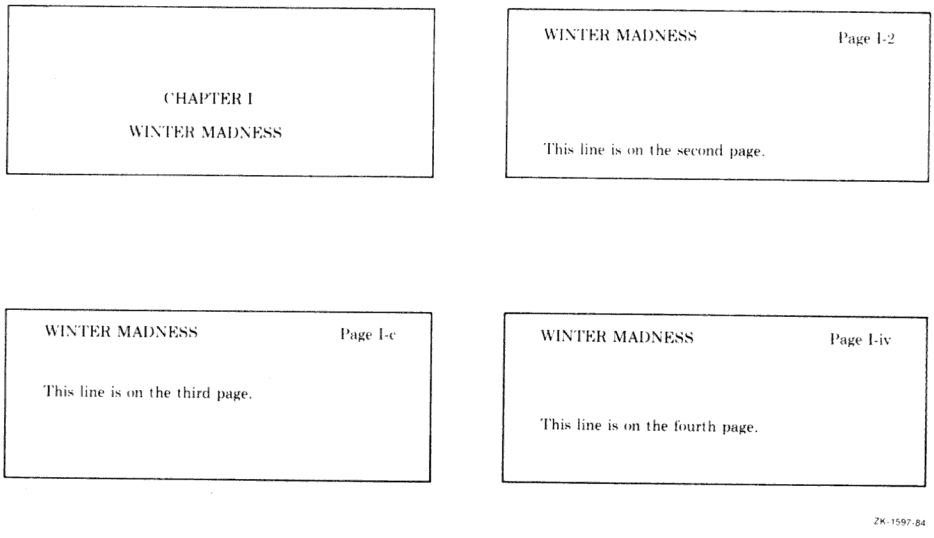
❶ This command numbers chapters with uppercase roman numerals (RU).

❷ This command changes page numbers from decimal numbers to lowercase letters (LL).

❸ This command changes page numbers from lowercase letters (LL) to lowercase roman numerals (RL).

Figure 4–9 shows the output file (CHAP.MEM).

Figure 4–9 Using the .DISPLAY NUMBER Command



4.9 How to Create an Appendix

You can use the .APPENDIX command to specify the beginning of an appendix. DSR will assign an identifying letter to it and allow you to supply a title. Successive .APPENDIX commands assign identifying letters in alphabetical order.

The syntax follows:

```
.APPENDIX text
```

"Text" indicates the title you give the appendix.

The following example shows how to create three consecutive appendixes.

Input file (APPENDIX.RNO):

```
This is the last line of text before the appendixes.
.APPENDIX First Title
.APPENDIX Second Title
.APPENDIX Third Title
```

Output file (APPENDIX.MEM):

```
This is the last line of text before the appendixes.


                              APPENDIX A
                             FIRST TITLE



                              APPENDIX B
                            SECOND TITLE



                              APPENDIX C
                             THIRD TITLE
```

4.10 How to Create Running Heads

By default, DSR provides page numbers at the top right of each page (except the first). The .HEADERS ON command controls DSR's ability to create running page numbers. Running page numbers, which are provided by default, are just part of the information that DSR is able to display about the contents of a page. By using the DSR commands discussed in the following sections, you can create one or two lines of information (running heads) at the top of each page.

If you do not want page numbers, you can disable the page numbering default with the .NO NUMBER command.

The following example shows how DSR formats pages, providing running page numbers by default:

Input file (RUNNING.RNO):

```
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number.
```

Output file (RUNNING.MEM):

```
This line of text is on the first page.  The first page does
not have a page number.

                                                        Page 2

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right.

                                                        Page 3

This line of text is on the third page.  Notice the position
of the page number.
```

4.10.1 Specifying a Title

If you want your running head information to contain a title, use the .TITLE command. By default, DSR displays this title at the top left of every page except the first.

The following example shows how DSR formats pages when you use the .TITLE command with running page numbers and a running title:

Input file (RUNNING.RNO):

```
.TITLE Whispering Willows
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number or a title.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right and a title at the top left.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number and the title.
```

Output file (RUNNING.MEM):

```
This line of text is on the first page.  The first page does
not have a page number or a title.

Whispering Willows                                    Page 2

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right and a title at the top left.

Whispering Willows                                    Page 3

This line of text is on the third page.  Notice the position
of the page number and the title.
```

4.10.2 Specifying the Date

When you want the current date to appear in running heads, use the .DATE command. The date appears in the format dd mm yy, for example, 14 December 1986, on the right side of the subtitle line. You must precede the .DATE command by the .SUBTITLE command.

The following example shows how DSR formats pages when you use the .DATE command with the .TITLE command:

Input file (RUNNING.RNO):

```
.TITLE Whispering Willows
.SUBTITLE
.DATE
.PAGE SIZE 15
.BLANK 15
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.
.BLANK 15
This line of text is on the second page.  From this point on,
every page has a number preceded by the word page at the top
right, a title at the top left, and the date.
.BLANK 15
This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

Output file (RUNNING.MEM):

```
This line of text is on the first page.  The first page does
not have a page number, a title, or the date.

Whispering Willows                                    Page 2
                                            14 December 1986

This line of text is on the second page.  From this point
on, every page has a number preceded by the word page at the
top right, a title at the top left, and the date.

Whispering Willows                                    Page 3
                                            14 December 1986

This line of text is on the third page.  Notice the position
of the page number, the title, and the date.
```

4.10.3 Specifying a Subtitle

When you want to specify a subtitle, use the .SUBTITLE command.

The following example shows how to use the .TITLE command and the .SUBTITLE command to create running heads with a title and a subtitle:

Input file (RUNNING.RNO):

```
.TITLE Faberge Fantasy
.SUBTITLE Easter Egg Gifts
.PAGE SIZE 15
.BLANK 15
The first page does not have any running head information.
.BLANK 15
The title and subtitle appear at the top left of the second page.
The page number is at the top right.
.BLANK 15
Notice the position of the title, subtitle, and page number on this page.
```

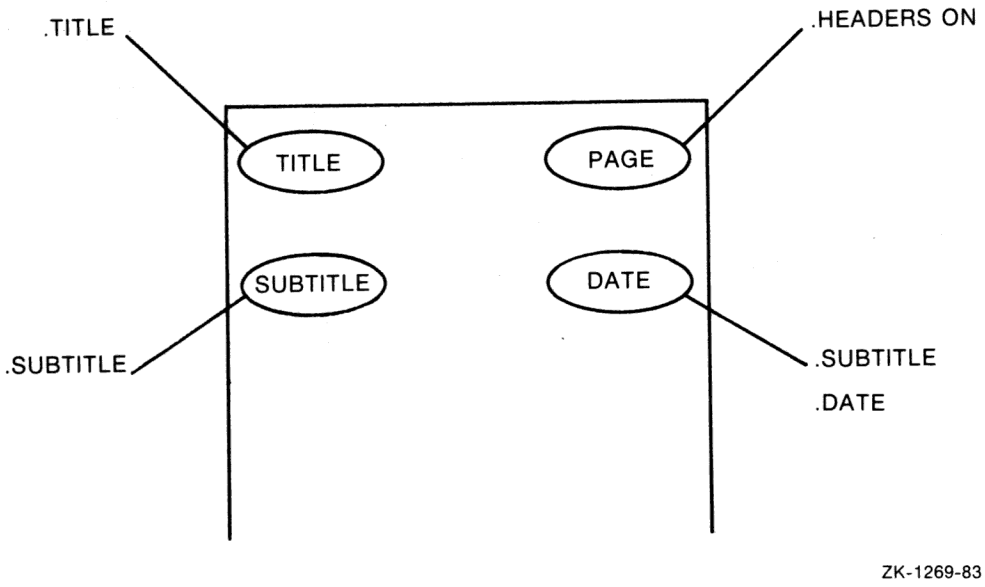
Output file (RUNNING.MEM):

```
The first page does not have any running head information.
Faberge Fantasy                                          Page 2
Easter Egg Gifts

The title and subtitle appear at the top left of the  second
page.  The page number is at the top right.
Faberge Fantasy                                          Page 3
Easter Egg Gifts

Notice the position of the title, subtitle, and page  number
on this page.
```

4.10.4 Organizing Running Head Information

Figure 4–10 shows the different parts of running head information and the commands you use to generate each part.

Figure 4–10 Running Head Information



You can use any combination of running head commands to alter the organization of information at the top of your documents.

4.10.5 Reorganizing Running Head Information

You can change the position of running head information on a page by using the .LAYOUT command. You can center titles and subtitles at the tops of pages, center page numbers at the bottom, and surround page numbers with hyphens. All the available options are shown in the following syntax description:

```
.LAYOUT n1,n2
```

The parameter "n1" is a number from 0 to 3 that specifies an alternative arrangement of running head information as follows:

#.LAYOUT 0	Restores the standard arrangement of a title and subtitle in the upper left of a page, and a page number and date in the upper right.
#.LAYOUT 1	Titles and subtitles are centered at the tops of pages. Page numbers are centered at the bottom.
#.LAYOUT 2	Titles and subtitles appear at the top right of right-hand (odd-numbered) pages and the top left of left-hand (even-numbered) pages. Page numbers are centered at the bottom.
#.LAYOUT 3	Gives the standard page arrangement for title and subtitle (as in .LAYOUT 0), but with the addition of running page numbers centered at the bottom of pages between two hyphens (for example, - 13 -). Running page numbers are consecutive throughout the entire document rather than within chapters.

The parameter "n2" specifies how many lines below the last line of text on a page the number will appear.

The following example uses the .LAYOUT 1 command, which centers the title and the subtitle at the top of the page and centers the page number at the bottom.

Input file (LAYOUT.RNO):

```
.TITLE Faberge Fantasy
.SUBTITLE Easter Egg Gifts
.LAYOUT 1,2
.PAGE SIZE 15
.BLANK 15
This line of text is on page 1.
.BLANK 15
This line of text is on page 2.
.BLANK 12
This line of text is on page 3.
.BLANK 10
This line of text is on page 4.
```

Output file (LAYOUT.MEM):

```
This line of text is on page 1.


                        1
                    Faberge Fantasy
                    Easter Egg Gifts

This line of text is on page 2.


                        2
                    Faberge Fantasy
                    Easter Egg Gifts

This line of text is on page 3.


                        3
                    Faberge Fantasy
                    Easter Egg Gifts

This line of text is on page 4.


                        4
```

For more information about the .LAYOUT command, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

4.10.6 Specifying the Title on the First Page

When you want running head information to appear on the first page of a document, use the .FIRST TITLE command. You must insert the .FIRST TITLE command before any text on the first page.

By default, DSR will use .HEADER LEVEL titles for running head subtitles. So, if you want your own subtitles to override the header level titles, you must use the .NO AUTOSUBTITLE command. For more information about the .AUTOSUBTITLE and .NO AUTOSUBTITLE commands, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

The following example shows how to create running head information on the first page by using the .FIRST TITLE command. The .NO AUTOSUBTITLE command causes the specified subtitle (Easter Egg Gifts) to override the header level title (Lilies of the Valley Egg).

(The text in the following example is used with permission from the publisher.)

Input file (RUNNING.RNO):

```
.FIRST TITLE
.TITLE Faberge Fantasy
.NO AUTOSUBTITLE
.SUBTITLE Easter Egg Gifts
.DATE
.HEADER LEVEL1Lilies of the Valley Egg
"This confection of pale pink enamel and pearls is one of the most
original of Faberge's creations and, with the Pansy Egg of 1899, one
of only two Imperial eggs executed in the then-fashionable Art Nouveau style."
.HEADER LEVEL2Materials
"The egg is supported on cabriole legs of matte green-gold leaves dripping
with rose diamond dewdrops.  Nestled in a bouquet of gold-stemmed pearl and
diamond lilies of the valley with translucent green enamel leaves, the pink
egg is surmounted by a miniature Imperial crown of rose diamonds and
cabochon rubies."
.HEADER LEVEL3Height
The height of the egg is 5 and 5/16 inches when unopened and 7 and 7/8 inches
when opened.
.HEADER LEVEL2
"A small pearl knob triggers the surprise - a trefoil of Zehngraf's portrait
miniatures of Czar Nicholas II and his eldest daughters, Grand Duchesses
Olga and Tatiana.  A geared mechanism raises the miniatures, which spread
fanlike upon emerging from within the egg."
```

Output file (RUNNING.MEM):

```
Faberge Fantasy                                        Page 1
Easter Egg Gifts                              14 December 1986

1  LILIES OF THE VALLEY EGG

"This confection of pale pink enamel and pearls is  one  of  the  most
original  of  Faberge's creations and, with the Pansy Egg of 1899, one
of only two Imperial eggs executed in the then-fashionable Art Nouveau
style."

1.1  Materials

"The egg is supported on cabriole  legs  of  matte  green-gold  leaves
dripping  with  rose  diamond  dewdrops.  Nestled  in  a  bouquet  of
gold-stemmed pearl and diamond lilies of the valley  with  translucent
green  enamel  leaves,  the  pink  egg  is  surmounted  by a miniature
Imperial crown of rose diamonds and cabochon rubies."

1.1.1  Height - The height of the  egg  is  5  and  5/16  inches  when
unopened and 7 and 7/8 inches when opened.

1.2

"A small pearl knob triggers the surprise - a  trefoil  of  Zehngraf's
portrait  miniatures  of  Czar  Nicholas  II and his eldest daughters,
Grand Duchesses Olga and  Tatiana.  A  geared  mechanism  raises  the
miniatures, which spread fanlike upon emerging from within the egg."
```

4.11 How to Create Notes and Footnotes

You can use DSR to make text stand out on a page to format notes and footnotes. This section discusses the DSR commands you need to use to create notes and footnotes. Notes are discussed first, then footnotes.

4.11.1 Using the .NOTE Command

The .NOTE command causes DSR to make the margins narrower, center a title over the text, and leave two blank lines before and one blank line after the title. The .END NOTE command causes DSR to leave a blank line after the note and restore the margin settings that were in effect before you issued .NOTE.

The syntax for this command is:

```
.NOTE text
```

If you do not specify a title for the note, DSR provides the word NOTE.

The following example demonstrates how to create a note using the .NOTE command.

Input file (NOTE.RNO):

```
When you are entering text and you want to set off some
information, you can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.

.NOTE Note Fun
This text is part of the note.  The margins are much narrower.
There are two blank lines before and one after the note.  Also,
the title is centered over the note.
.END NOTE
When you issue the .END NOTE command, your original margins
return.
```

Output file (NOTE.MEM):

```
When you are entering text and you want  to  set  off  some
information,  you  can use the .NOTE and .END NOTE commands.
Notice the title and change in margins that DSR provides.

                          Note Fun

               This text is part of the note.
               The margins are much narrower.
               There  are  two  blank   lines
               before and one after the note.
               Also, the  title  is  centered
               over the note.

When you issue the .END NOTE command, your original  margins
return.
```

4.11.2 Using the .FOOTNOTE Command

You can use the .FOOTNOTE and .END FOOTNOTE commands to create footnotes. The .FOOTNOTE command places the text following it at the bottom of the current page if there is room. If there is not enough room for the entire footnote, DSR places it at the bottom of the next page.

The .END FOOTNOTE command ends the footnote and restores any case, fill, justify, spacing, or margin settings that you might have changed in the footnote text.

The following example demonstrates how to create footnotes:

Input file (FOOT.RNO):

```
.PAGE SIZE 35,55
When you want to add a footnote to your text, use the .FOOTNOTE
and .END FOOTNOTE commands.  DSR puts the footnote at the bottom
of the page.  If there is not enough room on the current page,
DSR puts the footnote on the next page.
.BLANK
For this example, the page size is set to 35 lines long and 55
characters wide.
.FOOTNOTE
This text is part of the footnote.  Notice where it appears on
the page.
.END FOOTNOTE
```

Output file (FOOT.MEM):

```
When you want to add a footnote to your text, use  the
.FOOTNOTE  and  .END  FOOTNOTE  commands.  DSR puts the
footnote at the bottom of the page.  If there  is  not
enough room on the current page, DSR puts the footnote
on the next page.

For this example, the page size is set to 35 lines long
and 55 characters wide.




This text is part of the footnote.   Notice  where  it
appears on the page.
```

The .FOOTNOTE command does not provide a footnote symbol such as an asterisk (*) or (1). You can add one of these symbols before the text of the footnote by adding the .LEFT MARGIN and .INDENT commands as the following syntax shows:

Using an asterisk:

```
.FOOTNOTE
.LEFT MARGIN -2;*#text
.END FOOTNOTE
```

Using a number:

```
.FOOTNOTE
.LEFT MARGIN -2;(1)#text
.END FOOTNOTE
```

"Text" is the text of the footnote.

The following example shows how DSR moves the footnote to the next page and how you can precede the footnote by an asterisk.

Input file (FOOT.RNO):

```
.PAGE SIZE 15
This is the first line of text on this page.
Each page will have 15 lines.  If the footnote does not fit
on the page on which it occurs, DSR places it on the bottom
of the next page. But, DSR will not split a single footnote
over two pages.
.BLANK 2
The .FOOTNOTE command does not provide a footnote symbol such
as an asterisk (*) or (1).  But, if you want to put an asterisk
before the text of the footnote, add the .LEFT MARGIN command
and the .INDENT command.
.FOOTNOTE
.LEFT MARGIN 2
.INDENT -2;*#
This is the first line of the footnote.  Because there is not
enough room on the first page, DSR will put this footnote on
the second page.
.END FOOTNOTE
.BLANK 3
Here is more text following the footnote insert.
```

Output file (FOOT.MEM):

```
This is the first line of text on this page.  Each page will
have 15 lines.  If the footnote does not fit on the page on
which it occurs, DSR places it on the bottom of the
next page.  But, DSR will not split a single footnote over
two pages.

The .FOOTNOTE command does not provide a footnote symbol
such as an asterisk (*) or (1).  But, if you want to put an
asterisk before the text of the footnote, add the .LEFT
MARGIN command and the .INDENT command.
```

```
                                                        Page 2

Here is more text following the footnote insert.


* This is the first line of the footnote.  Because there is
  not  enough  room  on  the  first  page, DSR will put this
  footnote on the second page.
```

For more information about the .NOTE and .FOOTNOTE commands, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

4.12 How to Emphasize Text

DSR allows you to emphasize text by either boldfacing or underlining. To boldface or underline, you must use a special character called a flag. (To boldface, you also need to enter the .FLAGS BOLD command, which causes DSR to recognize the bold flag.) You insert these flags into your text where you want the boldfacing or underlining to occur. The Bold and Underline flags follow:

```
Bold Flag          (*)
Underline Flag     (&)
```

When you precede a character by the Bold flag (*), the character is boldfaced, that is, overstruck once. You can cause the characters to be overstruck more than once by using the /BOLD qualifier in the DSR command line. (See the *VAX DIGITAL Standard Runoff (DSR) Reference Manual* for information about the /BOLD qualifier.)

The following example shows how to boldface individual characters.

Input file (BOLD.RNO):

```
.FLAGS BOLD
Follow route *3 to route *7.
```

Output file (BOLD.MEM):

```
Follow route 3 to route 7.
```

You can pair the Bold flag with the Uppercase flag (^*) to turn boldfacing on and pair it with the Lowercase flag (*) to turn boldfacing off. The following example demonstrates how to boldface an entire line of text.

Input file (BOLD.RNO):

```
.FLAGS BOLD
^*This entire line of text is in boldface.\*
```

Output file (BOLD.MEM):

```
This entire line of text is in boldface.
```

When you precede a character by the Underline flag (&), DSR underlines the character. The following example shows how to underline individual characters.

Input file (UNDERLINE.RNO):

```
&A is for Amy and &B is for Basil...
```

Output file (UNDERLINE.MEM):

```
A is for Amy and B is for Basil...
```

You can pair the Underline flag with the Uppercase flag (^&) to turn underlining on and pair it with the Lowercase flag (\&) to turn underlining off. The following example demonstrates how to underline an entire line of text.

Input file (UNDERLINE.RNO):

```
^&KEEP OFF THE GRASS, PLEASE\&
```

Output file (UNDERLINE.MEM):

```
KEEP OFF THE GRASS, PLEASE
```

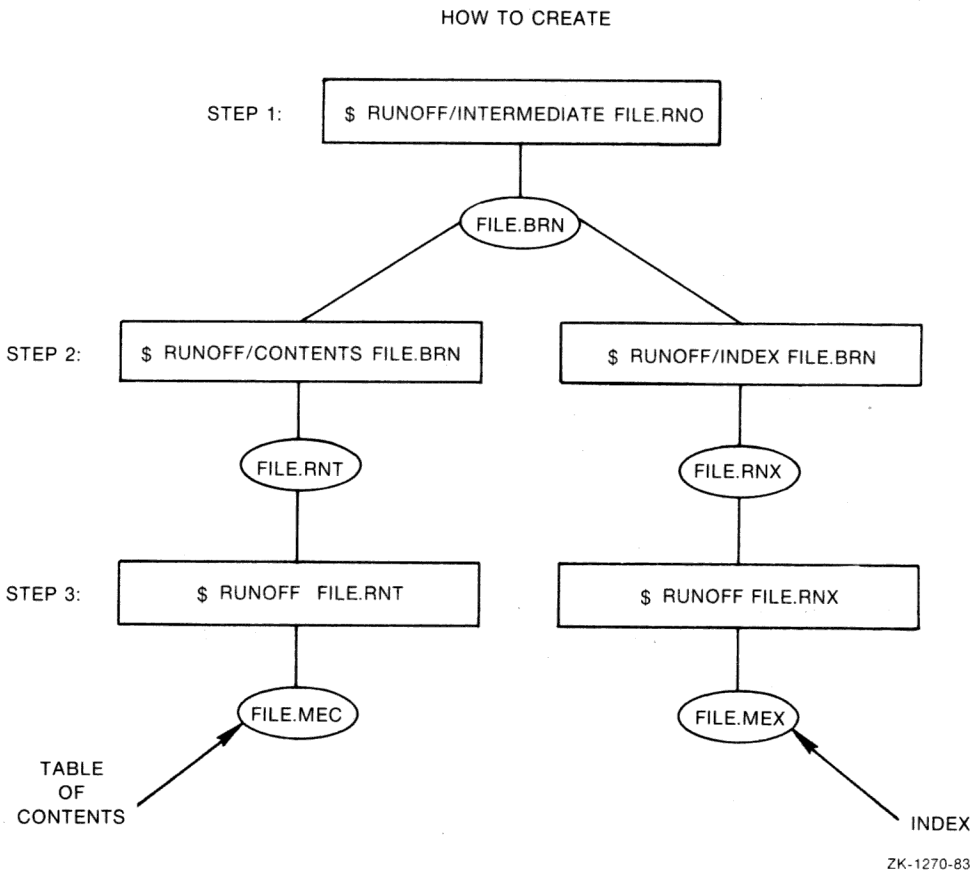
4.13 How to Create a Table of Contents and an Index

You can use DSR to create a table of contents or an index. The table of contents you produce can display chapter titles and numbers, header levels, and appendix titles and letters. The index you produce can display a two-column index with alphabetized entries at the left of each column. Each entry is separated from its page numbers by a comma. Entries with different first letters are separated by a blank line.

Figure 4–11 displays the three steps you follow to create either a table of contents or an index.

As Figure 4–11 shows, Step 1 is the same whether you are creating a table of contents or an index. Step 1 produces a single intermediate (binary) file, with a file type of BRN. This BRN file contains both table of contents and indexing information.

Figure 4–11 Creating a Table of Contents or an Index



4.13.1 Creating a Table of Contents

To create a table of contents, follow three steps:

1 Enter the following command line to generate an intermediate (binary) file:

```
$ RUNOFF/INTERMEDIATE file.RNO
```

Ensure that you specify an RNO file. DSR produces a BRN file.

2 Enter the following command line to run the table of contents program:

```
$ RUNOFF/CONTENTS file.BRN
```

Ensure that you specify a BRN file. You can add qualifiers to this command line to customize the table of contents program. DSR produces an RNT file.

3 Enter the following command line to process the RNT file:

```
$ RUNOFF FILE.RNT
```

Ensure that you specify an RNT file. DSR produces a MEC file. This MEC file contains the table of contents.

The following table shows the command lines you enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	.BRN
RUNOFF/CONTENTS FILE.BRN	.RNT
RUNOFF FILE.RNT	.MEC

When you enter the RUNOFF/CONTENTS command without qualifiers, you get the following defaults:

- Chapter titles and numbers (generated by the .CHAPTER command)
- Section titles and numbers (generated by the .HEADER LEVEL command). By default, DSR allows up to six levels of headers to appear in the table of contents.
- Appendix titles and letters (generated by the .APPENDIX command)
- Chapter-oriented page numbers (1-1, 1-2, 1-3, . . .) for all table of contents entries.
- An output file with the same name as the input file.

4.13.1.1 Tailoring the Table of Contents Program

To tailor the DSR table of contents program to your own needs, use the qualifiers listed in the following table.

Qualifier	Results
/BOLD	Any bolding specified in chapter and header titles appears in the table of contents.
/DEEPEST_HEADER=n	The number you specify for n determines the deepest header level displayed in the table of contents.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of RNT.
/LOG	Processing information is displayed.
/IDENTIFICATION	The version of DSR used to process your file is displayed.
/NOOUTPUT	An output file is not produced.
/PAGE_NUMBERS=RUNNING	Running page numbers appear instead of chapter-oriented page numbers for all table of contents entries, whether or not you specified running page numbers in the document.
/NOSECTION_NUMBERS	Header level numbers do not appear in the table of contents.
/UNDERLINE	Any underlining specified in chapter and header titles appears in the table of contents.

4.13.1.2 Looking at Tables of Contents

The following examples demonstrate how two of the qualifiers already described can alter the appearance of a table of contents. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual*.

The first example shows a table of contents produced by DSR defaults. No qualifiers are added to the command line:

```
$ RUNOFF/CONTENTS
```

```
                                CONTENTS

CHAPTER 1       HOW TO TILE A FLOOR
        1.1     READING ABOUT TILING . . . . . 1-1
        1.1.1     Tiling For Fun . . . . . 1-2
        1.1.2     Your Home In Tile . . . . . 1-3
        1.1.3     Changing A Room With Tile . . . . . 1-3
        1.1.4     How To Tile Floors And Walls . . . . . 1-4
        1.2     BUYING THE TILE . . . . . 1-5
        1.2.1     Researching Tiles Produced Abroad . . 1-5
        1.2.2     Coordinating Colors . . . . . 1-6
        1.2.2.1   Colors That Fade . . . . . 1-6
        1.2.3     Tile Textures . . . . . 1-6
        1.2.4     Types Of Tiles . . . . . 1-7
        1.2.4.1   Ceramic . . . . . 1-7
        1.2.4.2   Clay . . . . . 1-7
        1.2.4.3   Cement . . . . . 1-7
        1.3     TOOLS FOR TILES . . . . . 1-8
        1.3.1     Renting A Cutter . . . . . 1-8
        1.3.2     Buying Or Renting Crimpers . . . . . 1-9
        1.4     ACCOMPANYING MATERIALS . . . . . 1-9
        1.4.1     How To Adhere The Tiles . . . . . 1-9
        1.4.1.1   Mastic . . . . . 1-9
        1.4.1.1.1 Types Of Mastic . . . . . 1-10
        1.4.2     Grout . . . . . 1-10
        1.4.2.1   Coordinating Colors . . . . . 1-10
        1.4.2.2   How To Mix . . . . . 1-11
        1.4.2.3   How To Apply . . . . . 1-11

CHAPTER 2       HOW TO CEDAR A CEILING
        2.1     GETTING STARTED . . . . . 2-1
        2.1.1     Various Surfaces . . . . . 2-1
```

The second example shows how to change the display of page numbers from chapter-oriented numbers (1-1, 1-2, 1-3, . . .) to running numbers (1,2,3, . . .). The command line is:

```
$ RUNOFF/CONTENTS/PAGE_NUMBERS=RUNNING
```

```
                                CONTENTS

CHAPTER 1       HOW TO TILE A FLOOR
        1.1     READING ABOUT TILING . . . . . 1
        1.1.1     Tiling For Fun . . . . . 2
        1.1.2     Your Home In Tile . . . . . 3
        1.1.3     Changing A Room With Tile . . . . . 3
        1.1.4     How To Tile Floors And Walls . . . . . 4
```

The third example shows how to display a table of contents without section numbers. The command line is:

```
$ RUNOFF/CONTENTS/NOSECTION_NUMBERS
```

CONTENTS

```
                ACCOMPANYING MATERIALS . . . . . 1-9
                  How To Adhere The Tiles . . . . . 1-9
                  Mastic . . . . . 1-9
                  Types Of Mastic . . . . . 1-10
                  Grout . . . . . 1-10
                  Coordinating Colors . . . . . 1-10
                  How To Mix . . . . . 1-11
                  How To Apply . . . . . 1-11

CHAPTER 2       HOW TO CEDAR A CEILING

                GETTING STARTED . . . . . 2-1
                  Various Surfaces . . . . . 2-1
```

4.13.1.3 Comparing New DSR with Previous Versions of DSR

With previous versions of DSR, you also followed three steps to create a table of contents. These three steps are shown in the following list beside their corresponding new steps:

1	OLD	RUNOFF/CONTENTS FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TOC (.BTC)
	NEW	RUNOFF/CONTENTS FILE.BRN
3	OLD	RUNOFF FILE.RNT
	NEW	RUNOFF FILE.RNT

As shown in the table, previous versions of DSR generated BTC files instead of BRN files. Even though these BTC files are no longer produced when you use the new DSR contents program, the new contents program will process any BTC files you still may have and want processed.

4.13.2 Creating an Index

To create an index, you enter .INDEX and .ENTRY commands throughout your file. The syntax for an index entry is:

```
.INDEX topic> subtopic> subtopic
or
.ENTRY topic
```

For example, if you want the word "clown fish" to appear in your index, you enter the .INDEX command followed by the word "clown fish:"

```
The tank was teaming with tetras, but Marvin was interested in
.INDEX clown fish
the clown fish at the front of the store.
```

For more information about the index commands, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

After you enter the index commands to your file, you are ready to run the indexing program. To run the indexing program, follow three steps:

1 Enter the following command line to generate an intermediate (binary) file:

```
$ RUNOFF/INTERMEDIATE FILE.RNO
```

Ensure that you specify a RNO file. DSR produces a BRN file.

2 Enter the following command line to run the indexing program:

```
$ RUNOFF/INDEX FILE.BRN
```

Ensure that you specify a BRN file. You can add qualifiers to this command line to customize the indexing program. DSR produces a .RNX file.

3 Enter the following command line to process the .RNX file:

```
$ RUNOFF FILE.RNX
```

Ensure that you specify an RNX file. DSR produces a MEX file. This MEX file contains the index.

The following table shows the command lines you enter for each step and the resulting file types:

Command Line	Resulting File Types
RUNOFF/INTERMEDIATE FILE.RNO	.BRN
RUNOFF/INDEX FILE.BRN	.RNX
RUNOFF FILE.RNX	.MEX

4.13.2.1 Tailoring the Index Program

When you use the RUNOFF/INDEX command without qualifiers, you get the following defaults:

- 55 lines per page, including the top and bottom header areas
- Chapter-oriented page numbers for index entries
- Consecutive page numbers merged into ranges
- An output file with the same name as the input file

You can use the qualifiers listed in the following table to tailor the indexing program.

Qualifier	Results
/IDENTIFICATION	The version of DSR used to process your file is displayed.
/LINES_PER_PAGE=n	The number you specify for n determines the size of the page text, including the top and bottom header areas.
/LOG	Processing information is displayed.
/OUTPUT=newfile	DSR produces an output file with the name specified by =newfile. This output file, like the default output file, has a file type of .RNX.
/NOOUTPUT	An output file is not produced.
/NOPAGE_NUMBERS	Index entries will not have page numbers.
/PAGE_NUMBERS=RUNNING	Running page numbers (1,2,3, . . .) appear instead of chapter-oriented page numbers (1-1, 1-2, 1-3, . . .) for all index entries, whether or not you specified running page numbers in the document.
/REQUIRE=filename	Allows you to change the heading on the first page of an index.
/RESERVE=n	DSR reserves n number of lines on the top of the index page.

For more information about all the available qualifiers you can use to create an index, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

4.13.2.2 Looking at Indexes

The following examples demonstrate how two of the qualifiers already described can alter the appearance of an index. For more information about these and other qualifiers, see the *VAX DIGITAL Standard Runoff (DSR) Reference Manual.*

The first example shows an index produced by DSR defaults. No qualifiers are added to the command line:

```
$ RUNOFF/INDEX
```

```
                                                        Page Index-1

                                INDEX

Amadeus                                     Liszt, franz, 3-2, 4-11
  see mozart
                                            Mozart, wolfgang amadeus, 3-5,
Bach, carl phillip emanuel, 1-2                 4-14
    to 1-3, 4-9
Bach, johann sebastian, 1-1, 3-2,           Prokofiev, sergei, 4-5, 4-15
    4-9, 4-12
Baroque composer                            Rachmaninoff, sergei, 3-3 to 3-4,
  see bach                                      4-13
Bartok, bela, 2-1, 3-4, 4-10,               Rite of spring
    4-13                                      see stravinsky
Britten, benjamin, 4-3, 4-14
                                            Satie, erik, 2-2, 4-10
Ceremony of carols                          Stravinsky, igor, 4-7, 4-15
  see britten                               Syrinx
Chopin, frederic, 4-3 to 4-4,                 debussy, 4-8, 4-17
    4-14
Debussy, claude, 3-3, 4-13                  Velvet gentleman
                                              see satie
French composer
  see debussy

                                            Waltz
Hindemith, paul, 4-6 to 4-7, 4-15             see chopin
```

The second example shows how to make DSR display index pages 15 lines long instead of 55 lines long (the default). The command line is:

```
$ RUNOFF/INDEX/LINES_PER_PAGE=15
```

```
                                                        Page Index-1

                                INDEX

Amadeus                                     Britten, benjamin, 4-3, 4-14
  see mozart
                                            Ceremony of carols
Bach, carl phillip emanuel, 1-2               see britten
    to 1-3, 4-9                             Chopin, frederic, 4-3 to 4-4,
Bach, johann sebastian, 1-1, 3-2,               4-14
    4-9, 4-12
Baroque composer                            Debussy, claude, 3-3, 4-13
  see bach
Bartok, bela, 2-1, 3-4, 4-10,               French composer
    4-13                                      see debussy
```

```
                                                        Page Index-2

Hindemith, paul, 4-6 to 4-7, 4-15
Liszt, franz, 3-2, 4-11

Mozart, wolfgang amadeus, 3-5,
    4-14
Prokofiev, sergei, 4-5, 4-15

Rachmaninoff, sergei, 3-3 to 3-4,
    4-13
Rite of spring
  see stravinsky
Satie, erik, 2-2, 4-10
Stravinsky, igor, 4-7, 4-15
Syrinx
  debussy, 4-8, 4-17
Velvet gentleman
  see satie

Waltz
  see chopin
```

The third example shows how to create an index with running page numbers (1,2,3, . . .) instead of chapter-oriented page numbers (1-1, 1-2, 1-3, . . .). The command line is:

```
$ RUNOFF/INDEX/PAGE_NUMBERS=RUNNING
```

```
                                                        Page Index-1

                                INDEX

Amadeus
  see mozart

Bach, carl phillip emanuel, 2 to
    3, 20
Bach, johann sebastian, 1, 8, 20,
    23
Baroque composer
  see bach
Bartok, bela, 4, 10, 21, 24
Britten, benjamin, 14, 25

Ceremony of carols
  see britten
Chopin, frederic, 14 to 15, 25
Debussy, claude, 9, 24

French composer
  see debussy

Hindemith, paul, 17 to 18, 26
Liszt, franz, 8, 22

Mozart, wolfgang amadeus, 11, 25

Prokofiev, sergei, 16, 26

Rachmaninoff, sergei, 9 to 10, 24
Rite of spring
  see stravinsky

Satie, erik, 5, 21
Stravinsky, igor, 18, 26
Syrinx
  debussy, 19, 28

Velvet gentleman
  see satie


Waltz
  see chopin
```

4.13.2.3 Comparing New DSR with Previous Versions of DSR

With previous versions of DSR, you also followed three steps to create an index. These three steps are shown in the following list, beside their corresponding new steps:

1	OLD	RUNOFF/INDEX FILE.RNO
	NEW	RUNOFF/INTERMEDIATE FILE.RNO
2	OLD	RUN SYS$SYSTEM:TCX (.BIX)
	NEW	RUNOFF/INDEX FILE.BRN
3	OLD	RUNOFF FILE.RNX
	NEW	RUNOFF FILE.RNX

As shown in the list, previous versions of DSR used to generate BIX files instead of BRN files. Even though these BIX files are no longer produced, the new indexing program will process any BIX files you still may have and want processed.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Guide to Text Processing on VAX/VMS
AI–Y502B–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line